Maxta Technology White Paper

Maximize the Promise of Hyper-Convergence

April 2017

Table of Contents

Introduction

The Maxta® solution is a hypervisor-agnostic, highly resilient storage platform for the virtual data center. Maxta provides its storage solution as the software-only Maxta Storage Platform (MxSP®) and as the validated Maxta MaxDeploy® Appliances. Maxta's software-centric, hyper-converged solution is transforming the enterprise storage market. It fully integrates with server virtualization at all levels, from the user interface to data management, and supports all possible deployments of virtual data centers, including private, public, and hybrid clouds. Maxta's software turns standard servers into a converged compute and storage solution, leveraging server-side flash and spinning disks to optimize performance and capacity. Maxta's distributed architecture enables shared storage with enterprise-class data services and full scale-out capability without performance degradation. This results in significant cost savings, as well as dramatically simplifying IT.

This paper will cover the evolution of the datacenter and discuss the technology Maxta has developed to address the challenges faced by the modern IT administrator.

Storage Challenges within the Software-Defined Data Center

The Software-Defined Data Center (SDDC) is a unified data center platform that delivers converged compute, storage, and networking. SDDC leverages commodity hardware and intelligent software to pool resources and manage them in a global manner for improved utilization and efficiency.

While server virtualization has created an efficient infrastructure by enabling aggregation of compute resources (CPU, memory), storage technologies have only made incremental innovations in the past decade. Shared storage, generally SAN or NAS, is a requirement for utilizing many of the benefits of server virtualization. However, these networked storage solutions are expensive and increase the complexity of storage management due to the mismatch between storage constructs (e.g., LUNs, volumes, file systems) and virtualization constructs (Virtual Machines or VMs). The advent of flash storage has significantly improved the performance of storage, but these solutions do not address the issues of cost and complexity.

In recent years, alternative approaches to networked storage, such as virtual storage appliance (VSA) and distributed file system, have emerged. However, these approaches are limited in terms of scalability and performance, while not improving storage management. Thus, they are not an alternative to networked storage in most cases. Other alternatives to networked storage such as Hadoop have emerged as well for Big Data environments and specific storage workloads. However, these alternatives are not well suited for general purpose virtualized environments and associated storage workloads.

What is Hyper-Convergence?

Hyper-convergence and Software-Defined Storage (SDS) address the challenges of storage in virtualized environments by implementing storage functionality entirely in software while leveraging commodity components. The software is deployed on the servers that are running the hypervisor and enables pooling of disperse server-side physical storage resources. SDS provides VM-centric management of the storage pool, thereby addressing the gap between storage constructs and virtualization constructs.

In this way, a well-defined hyper-converged infrastructure will deliver resource efficiency by utilizing industry standard servers, SSD, PCI-attached flash, and magnetic disk drives. It will provide operational efficiency by managing integrated software entities (converged compute and storage) rather than isolated hardware resources (storage arrays). A software-defined approach also allows for quick responses to support new requirements. Finally, there should be no need to coordinate or negotiate storage resources with a separate storage team and no need to forecast and manage different physical storage resource pools.

Maxta at a Glance

Technology

Maxta has developed a VM Storage Platform that enables IT to fully realize the vision of the virtual data center. MxSP addresses the issues discussed above with:

- Hypervisor-agnostic implementation currently supports VMware vSphere, KVM, and OpenStack
- Full integration into the server virtualization user interface
- Scale-up and scale-out compute and storage independently on demand
- VM-level storage abstraction rather than block or file level abstraction
- VM-centric enterprise-class data services
- Support live migration, dynamic load balancing, high availability, data protection, and disaster recovery
- Optimized for flash performance and hard disk capacity

In summary, MxSP significantly simplifies IT, increases IT efficiency, and dramatically reduces capital and operational expenditures.

Solutions

Maxta offers two different types of solutions:

- MaxDeploy Hyper-converged Appliance
- MxSP software-only

The MaxDeploy appliance delivers a flexible way of deploying hyper-converged solutions via predefined and pre-validated solutions that combine Maxta's software along with partner solutions and platforms. This removes interoperability and performance guesswork and simplifies the ordering process.

MxSP software-defined storage solutions provide customers the complete flexibility to customize their solutions and run on existing x86 servers. Maxta's server configuration tools and appliance briefs can help with configuration suggestions and compatibility verifications.

Architecture

Overview

MxSP's innovative, peer-to-peer architecture aggregates storage resources from multiple servers, incorporating a global namespace and creating a Maxta storage pool. An instance of MxSP software is installed on each of the servers that are part of the virtualization cluster. The software leverages flash storage resources such as PCIe, NVMe, or SATA SSDs for performance and SATA or SAS HDDs for capacity. All the servers running MxSP software have access to the aggregated Maxta storage pool. However, it is not a requirement that all servers have storage resources or contribute storage resources to the Maxta storage pool. It is also not a requirement that the servers that contribute storage resources to the Maxta storage pool contribute the same capacity. The communication between the MxSP instances is over a minimum 1GbE shared or dedicated Ethernet network. It is highly recommended that the network be a 10 GbE dedicated private network.

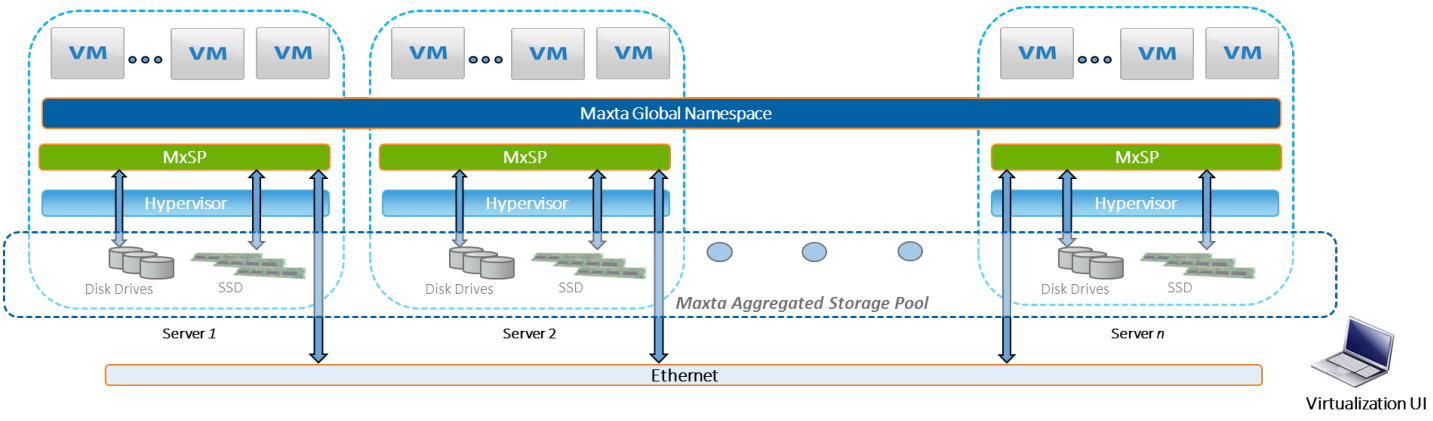


Figure 1: MxSP Architecture

Hypervisor Agnostic Architecture

MxSP is a hypervisor agnostic implementation of an enterprise storage solution for virtualized environments. The diagram below describes the architectural implementation.

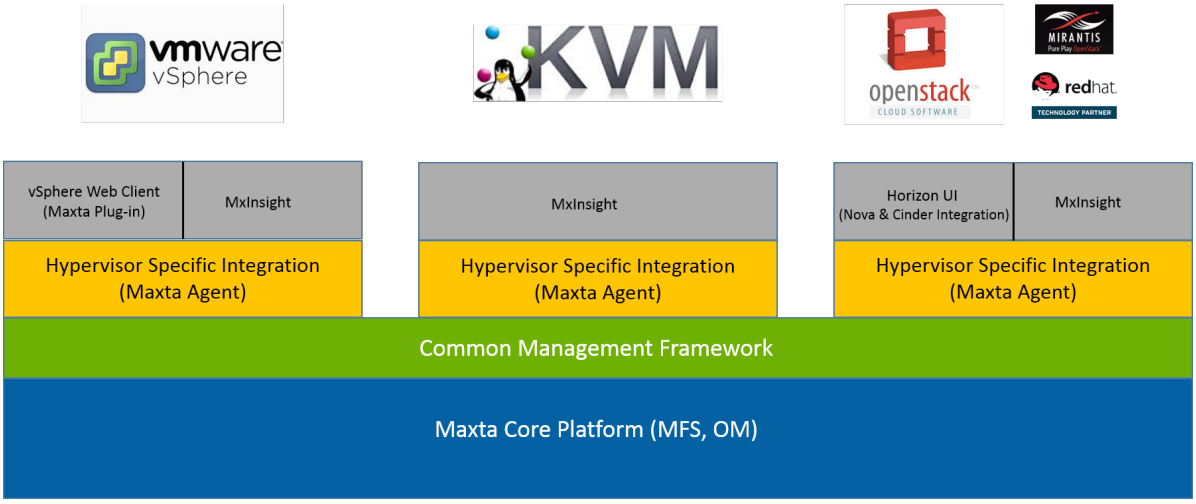


Figure 2: MxSP Hypervisor Agnostic Architecture

The Maxta Core Platform consists of the Maxta Distributed File System (MFS) and Object Manager (OM). The core platform and Maxta's Common Management Framework are common across all hypervisors. A Maxta agent is developed to integrate into each specific hypervisor. The same MxInsight® web management interface is available on all hypervisors. Additionally, Maxta has developed a plug-in into the VMware vSphere web client for MxInsight as well as seamless integration into OpenStack's Horizon UI.

Maxta Architecture in a VMware Environment

There are a few aspects of the Maxta architecture which have been optimized specifically for VMware environments. Maxta storage is presented as an NFS datastore shared across all ESXi hosts within a cluster. MxSP runs as a VM consuming 4 vCPU and 20 GB RAM, parameters which can be adjusted based on performance requirements. MxInsight is directly integrated into the vSphere Web Client as a legacy plug-in to provide full management capability from a single pane of glass. Maxta software is compatible with all versions of VMware vSphere 5.5 U2 or later.

Maxta Architecture in KVM and OpenStack Environments

MxSP provides the same functionality in KVM and OpenStack as is available for VMware environments. There are a few key differences in management and implementation which are important to point out here. One difference is that Maxta storage is presented as local storage accessible to each node in an OpenStack or KVM deployment. Additionally, MxSP runs as a process within the hypervisor.

Maxta has developed drivers for Cinder and Nova to plug directly into OpenStack's block storage and compute functionality. These plug-ins improve upon the existing OpenStack functionality in two key areas. First, administrators are able to create thin-provisioned instances and volumes directly on Maxta storage using OpenStack's APIs. Second, Maxta provides improved control and functionality in an OpenStack environment with time, performance, and capacity-efficient snapshots and clones at an instance- or volume-level, rather than using OpenStack's standard snapshots, which are simply full copies of data. All of these operations can be performed directly through the Horizon UI since Maxta's Cinder and Nova drivers will intercept and redirect the API calls accordingly. Additionally, the same MxInsight interface that exists for VMware deployments is available for KVM and OpenStack environments.

MxSP can be run in a KVM environment without OpenStack, and is also compatible with the Kilo and later releases of OpenStack. Maxta is set up for continuous integration to validate the Cinder and Nova drivers beginning with the OpenStack Liberty release.

MxSP Components and Services

The diagram below shows the components of MxSP and their relation with one another in a Maxta environment:

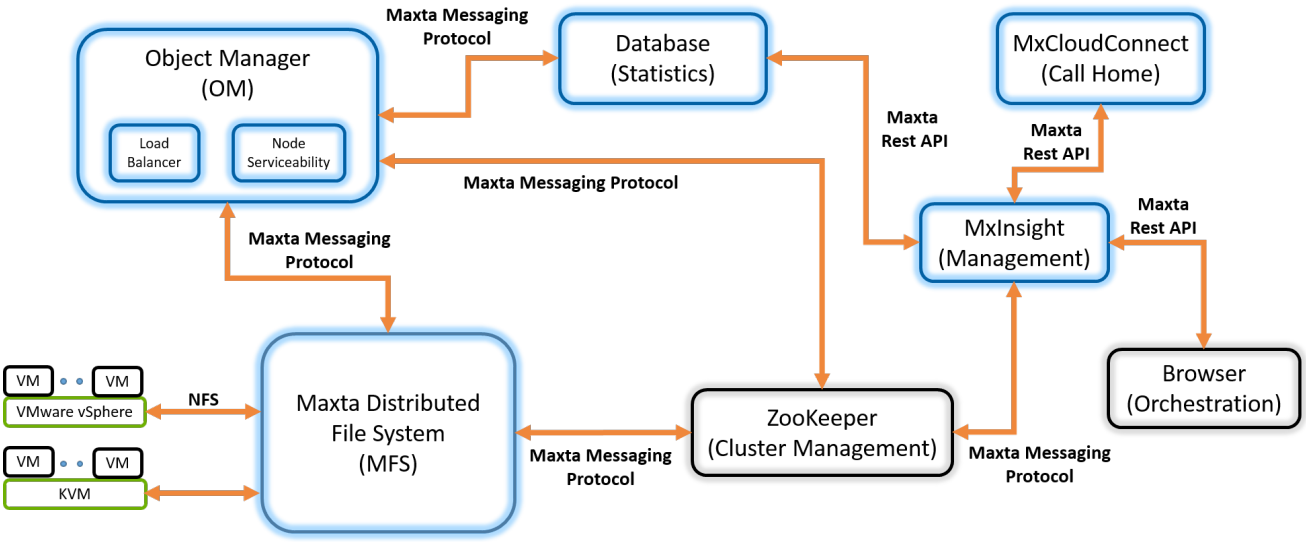


Figure 3: MxSP Component Relationship

MxSP core consists of four key components: Object Manager, Maxta Distributed File System, Database, and ZooKeeper. These components communicate with each using the Maxta Messaging Protocol. MxSP's management framework consists of three primary components (MxCloudConnect®, MxInsight, and the web browser), all of which communicate with MxSP core via Maxta REST API's. All components listed above are explained in further detail throughout this paper. MFS communicates with the hypervisor via NFS protocol.

Any instance of MxSP running in a virtualized environment consists of the following services:

- System Level
  - **monit** to monitor system status
  - **nfsd** for the NFS v3 server – only applicable to VMware environments
- Maxta File System (MFS)
  - **mfsd** for MFS
  - **mfsmonit** to monitor MFS
- Object Manager (OM)
  - **omNodeWatcher** to monitor node status
  - **omWalker** to initiate resync when necessary
- Management Framework
  - **Tomcat** to provide management API
  - **ZooKeeper** to provide clustering services

Clustering and Unified Namespace

MxSP presents shared storage at cluster-level visibility. The unified namespace service provides all nodes in the Maxta cluster with visibility to the same hierarchy of objects. This service also provides all nodes with the ability to modify any of these objects. The namespace remains globally consistent by ensuring that if two nodes want to modify the same set of data, only one of the nodes will be able to complete that operation. At any point in time, a minimum of three nodes must be configured to run the unified namespace service, and a majority of those nodes must be online to maintain quorum.

The ZooKeeper service used by MxSP provides synchronization and maintains a quorum across all nodes in the Maxta cluster. ZooKeeper and unified namespace services are deployed on all nodes in a Maxta cluster and these services are in an active state on a select number of nodes depending on cluster size. The ZooKeeper service also requires a minimum of three nodes, with a majority of the nodes online in order to maintain quorum. MxSP's high availability services ensure that data remains accessible when nodes fail and can provide greater fault tolerance depending on cluster size, replica policy, and fault containment units made available via Rack Awareness.

Maxta Distributed File System

MxSP uses Maxta's own highly scalable, resilient Maxta Distributed File System (MFS) to deliver shared storage with enterprise class services. MFS has been developed with the purpose of meeting the needs of an enterprise hyper-converged virtual environment. MFS eliminates split-brain conditions by synchronizing operations between instances across the cluster. MFS runs on each server in the virtualization cluster to deliver a highly available and scalable global namespace.

I/O Path

Maxta has optimized its I/O path to maintain high performance in virtualized environments which utilize a hybrid disk configuration consisting of spinning disks and flash media. Read and write operations are directed to the flash media or SSD before spinning disk, thereby maximizing performance in hybrid storage deployments. The pattern of read/write operations in a virtualized environment is predominately random in nature since the storage workloads of multiple virtual machines hosted on a virtualized server are blended together. Therefore, the storage workload always appears random, independent of the individual storage workloads of individual virtual machines. MxSP accelerates the random read/write operations by using a log based layout, with distributed metadata used to map data blocks to their storage locations.

Maxta leverages flash media for read and write-back caching. Enterprise MLC flash devices are used in MxSP deployments to guarantee that data is not lost in the case of power failure. Writes are directed to the flash media in a sequential manner and replicas are written to flash on other nodes in order to maintain data availability. The writes are later de-staged to magnetic drives in a sequential manner by collecting several thousand small random writes from the Maxta Intent Log (MxIL), allocating space sequentially on the magnetic disk, and then performing the write as a few large I/Os. Performing writes to flash and magnetic drives in a sequential manner maximizes I/O performance and SSD endurance.

For a complete description of how data is located within a Maxta cluster, see the Data Layout section.

A write I/O proceeds as follows:

1) A Maxta write I/O is processed through the local MxSP instance closest to the VM guest.
2) The data is striped as widely as possible and replica copies are created. The data and its replicas are written across the network to MxIL residing on flash media on at least two different nodes.
3) Each node acknowledges that data has been written to the flash media.
4) The write I/O is acknowledged back to the application after all data, including replicas, has been written to flash.
5) Data is eventually de-staged onto spinning disk.

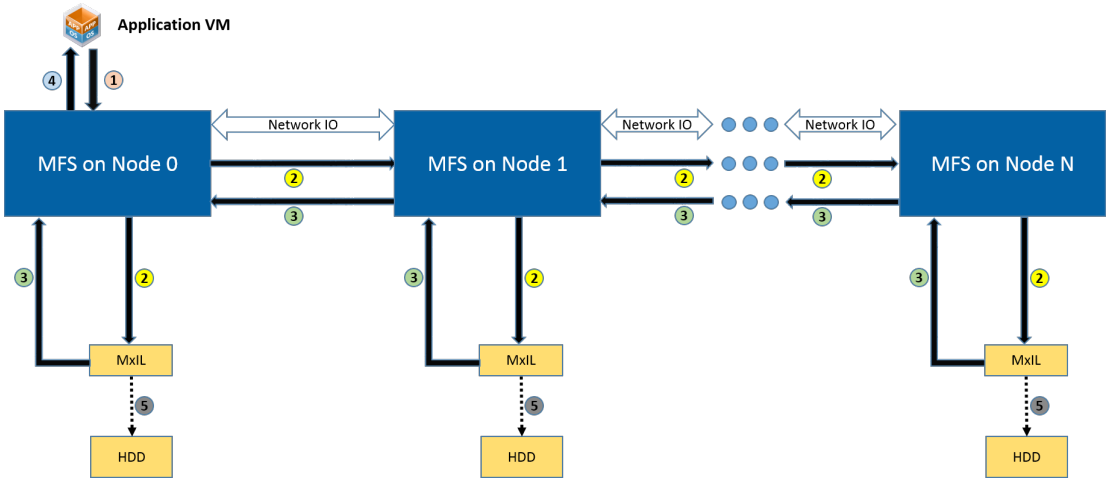


Figure 4: MxSP Write I/O Path

Maxta also implements various data layout policies such as Rack Awareness, Maxta Metro Storage Cluster, and Node Affinity. If any of these features are enabled when data is written, MxSP makes sure that a member and its replica abide to the data placement constraints of the specified policy.

Read operations are accelerated by maintaining metadata and frequently accessed data on the SSD cache. MFS also leverages RAM from its controller VM as a cache tier ahead of flash. However, flash is substantially larger than RAM and is therefore able to support larger working sets. While MFS has been optimized to deliver cost efficiency and high performance in hybrid storage configurations (mix of HDD and flash/SSD).

A read I/O proceeds as follows (note that if the original copy of data is marked as stale then the data is fetched from its replica copy):

1) A Maxta read I/O is processed through the local MxSP instance closest to the VM guest.
2) If the data is dirty (has not yet been moved from MxIL to spinning disk) then it is read directly from MxIL.
3) The next level of read caching is the MxSP memory.
4) If there is a cache miss from memory then the data is fetched from the read cache residing on flash media.
5) If the data is not found in flash, MxSP will read the data from HDD.
6) The read I/O is acknowledged back to the application VM after striped data has been read from all nodes.

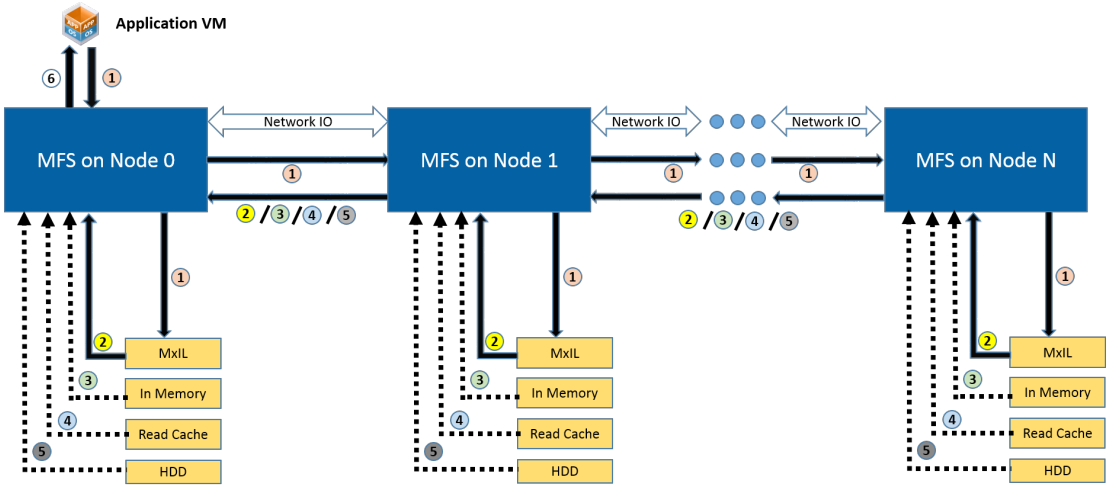


Figure 5: MxSP Read I/O Path

Floating Master

Traditional storage systems maintain a master for all VMs. This master is responsible for directing IOs between the VM and physical disks. The master exists in a server-client relationship and can become a bottleneck for IO traffic. Maxta has architected its master to be optimized for a VM-centric distributed file system. This is achieved by allowing the role of master to be passed from node to node and by maintaining masters on a per-VM basis.

Each controller VM within MxSP's distributed file system can assume the role of master for all guest VMs which reside on the same node as that controller. Similar to traditional storage systems, the controller VM which has the master role is responsible for directing IOs to different nodes across the cluster, as well as sending acknowledgments back to the VM itself. In order to eliminate unnecessary network hops in the case of a VM migration, MxSP has incorporated a floating master. In the case of a VM migration, the floating master architecture enables the controller VM on the destination node to take over the role of master after the guest VM is migrated.

For example, the workflow when a VM is first deployed to Node 0 in the cluster is:

1) The VM's master resides with the VM on Node 0.
2) Reads and writes come from the VM into Node 0.
3) The master directs these IOs across the network to the appropriate nodes.
4) The nodes send their acknowledgements back to the master on Node 0.
5) The master on Node 0 sends the acknowledgment to the VM.

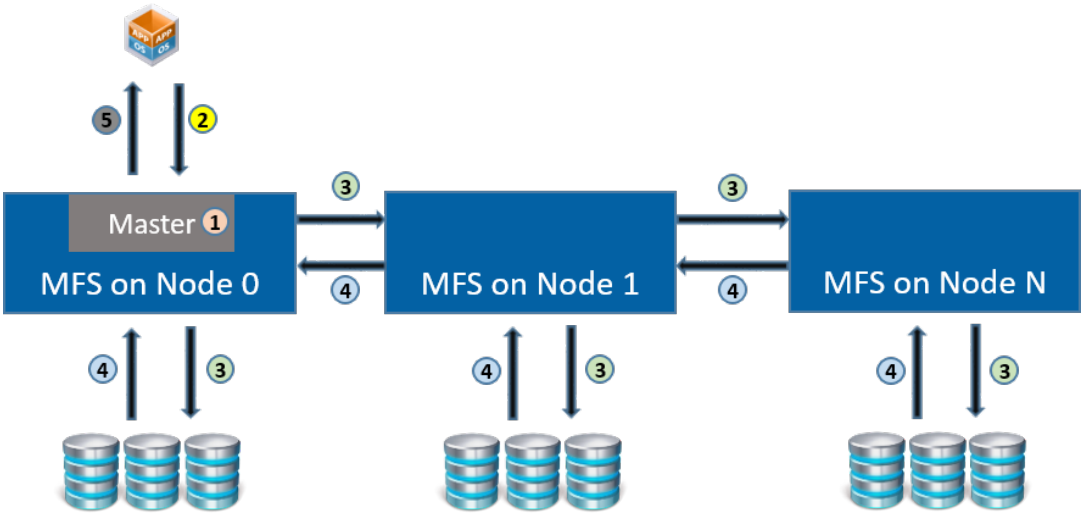


Figure 6: I/O Path with Floating Master after VM is deployed

Now, if that VM is migrated from Node 0 to Node 1 without MxSP's concept of a floating master:

1) The master remains on Node 0.
2) Reads and writes come from the VM into Node 1.
3) Node 1 sends the IOs to the master on Node 0.
4) The master directs these IOs across the network to the appropriate nodes.
5) The nodes send their acknowledgments back to the master on Node 0.
6) The master on Node 0 sends the acknowledgement to Node 1, where the VM resides.
7) Node 1 sends the acknowledgement to the VM.

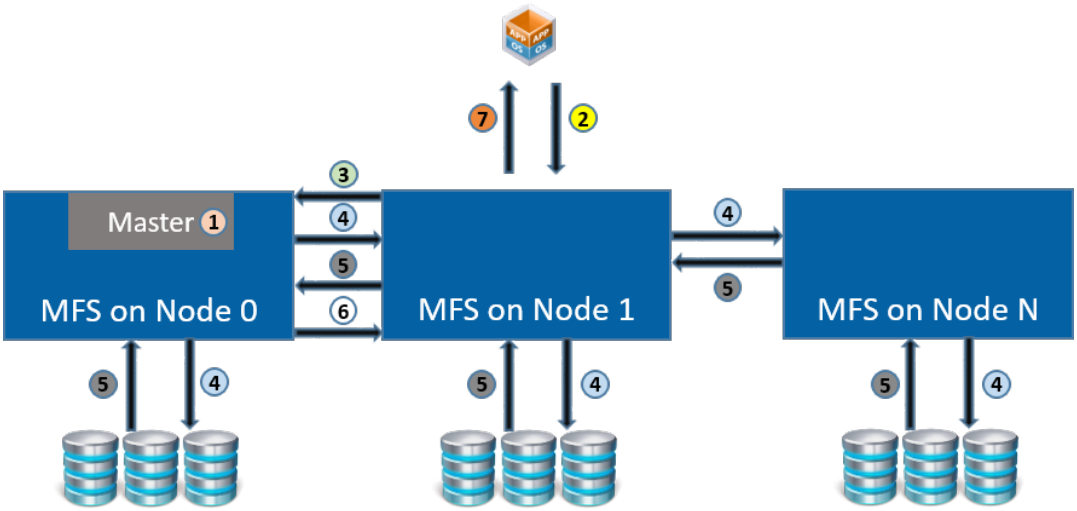


Figure 7: I/O Path with Static Master after VM is migrated

As shown in the above scenario, Step 3 and Step 6 are additional network hops that must happen when the master is not migrated with the VM.

With MxSP's floating master, after the VM has been migrated to Node 1:

1) The master moves with the VM to Node 1.
2) Reads and writes come from the VM into Node 1.
3) The master directs these IOs across the network to the appropriate nodes.
4) The nodes send their acknowledgments back to the master on Node 1.
5) The master on Node 1 sends the acknowledgment back to the VM.

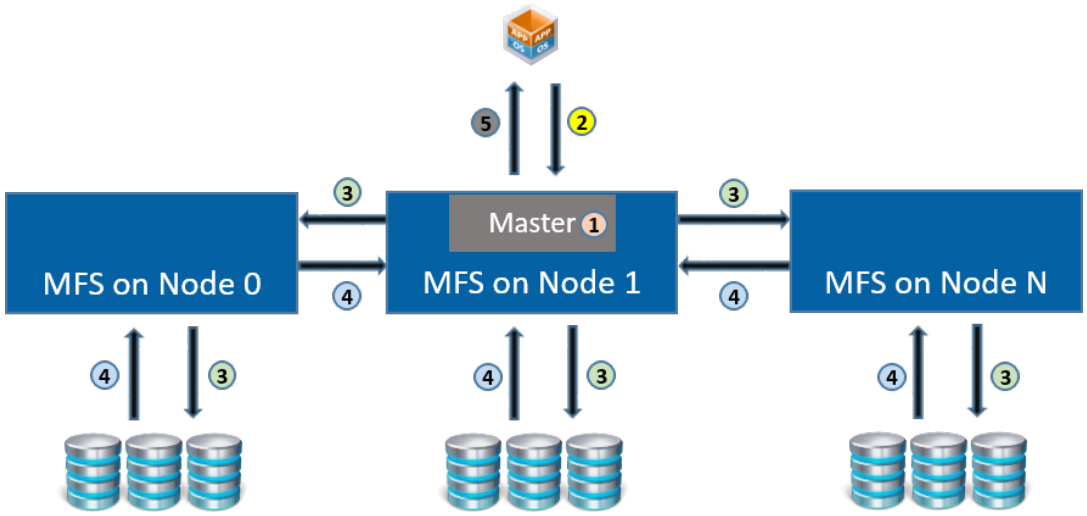


Figure 8: I/O Path with Floating Master after VM is migrated

When the floating master is implemented, as in the final scenario, there is no extra hop since the master has moved to the same node as the VM itself.

SSD Caching and Metadev

MxSP maximizes the performance to cost benefit of hybrid storage solutions by utilizing flash media for intent log, read cache, and flash accelerated metadata. Each server in a Maxta cluster is allocated 100+ GB for the intent log, 100+ GB for read cache, and at most 5% of HDD capacity for metadata. These numbers double in environments where SSD mirroring is enabled.

Maxta Metadev is a class of device in the Maxta file system. Metadev stores only the file system metadata to accelerate the read/write performance of metadata IOs. Maxta deployments typically use small page sizes (4K or 8K) for space efficiency and performance. Smaller page sizes would result in larger amounts of metadata and that creates random read/write IO performance issues since metadata cannot be cached entirely in memory.

Metadev is a way to separate metadata from data, allowing the metadata to be stored on high performance devices (like SSDs). This approach significantly accelerates the metadata read and write operations. Metadev stores metadata such as file indirect blocks and the file system's space allocation records, providing key information to the actual data itself.

The HDD to Metadev ratio depends specifically on the page size defined during the MxSP deployment. Deployments using a 4K page size have a recommended Metadev capacity of 5% of the node's HDD capacity and deployments using an 8K page size require 3% of the node's capacity.

MxSP uses a Shared Metadev format, wherein Metadev is created on SSDs that also host read/write-back cache partitions. In this way, users reduce costs by consolidating cache and Metadev onto the same SSD or flash device.

Maxta Intent Log

The Maxta Intent Log, or MxIL, resides on flash media and is the component of the Maxta File System primarily responsible for write transactions. MxIL serves as a large intent log which absorbs latency spikes that would otherwise occur during write operations. This is achieved by creating less metadata during random writes for larger working sets. MxIL eliminates the need to read and modify metadata during writes since the data is kept on the flash devices. After MxIL has collected several writes over a period of time, the data is eventually de-staged from MxIL to spinning disk in the background. The full read and write I/O paths are described more thoroughly in the I/O Path section above.

MxIL consumes 100 GB per node by default on the high-performing flash, but this amount is configurable and can be spread across up to four flash devices on each node. The large capacity intent log allows writes and overwrites of existing data to be kept in MxIL for longer without having to be flushed to the slower spinning disk. Data that is stored on MxIL is not compressed in order to improve latency for write operations; in-line compression occurs when data is de-staged to spinning disk. Additionally, Maxta has an option to mirror MxIL in order to provide redundancy and failover protection.

Data Services

MxSP delivers an unlimited number of time, performance, and capacity-efficient zero-copy snapshots and clones. These snapshots and clones are configured and managed at the VM level rather than at the storage level. This enables the VM administrator to leverage advanced storage features without the need for deep storage and vendor specific expertise.

Zero-copy Snapshots

MxSP leverages a log based layout approach for data placement and metadata for mapping data blocks to their storage locations. Therefore, once updated, the new image of a data block is not stored in-place to the same storage location that contains the previous image of the data block. Instead, the new image of the data block is written to a new storage location and the metadata is updated to reflect the change in the storage location of the data block. With this approach, the creation of a snapshot is as simple as marking the current file metadata read-only.

Snapshots are read-only, point-in-time virtual copies of data which do not consume any capacity and do not require any upfront space reservation. Snapshots are created in less than one second, independent of the capacity of the source, and snapshot creation does not impact the performance of virtual machines. Snapshots can be created even when a VM's replica data is not available due to an offline node. Once the node returns to an available state in the cluster, missed writes are replayed in order to the node. Snapshots are reconstructed and the node is once again fully re-synchronized.

Maxta's snapshots use a redirect-on-write methodology with no data movement. This enables snapshots to be deleted quickly and out of order since MFS simply deletes metadata which is no longer needed to reference the snapshot. Additionally, all snapshots on MxSP are crash consistent, providing the first layer of data protection without sacrificing performance or capacity.

Application Consistent Snapshots

MxSP also enables application consistent snapshots in environments that provide the appropriate tools for application consistent snapshots (e.g., VSS quiescing). This ensures that applications start from a consistent state when recovering a VM to a snapshot's point-in-time.

Snapshots can be taken through the Maxta UI, including the ability to select whether that snapshot should have application consistency.

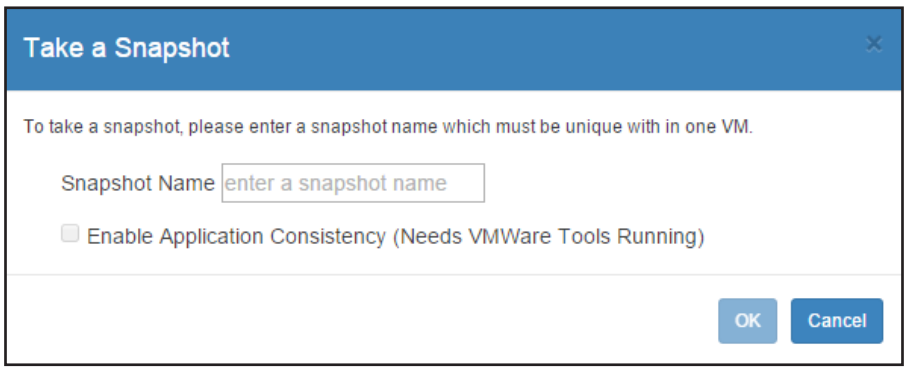


Figure 9: Application Consistent Snapshots via MxInsight

Zero-copy Clones

Maxta's zero-copy clones use the same methodology as Maxta's snapshots with the difference being that clones are read/write virtual copies of the data. Once a data block is updated by a VM or a clone, the metadata for the respective VM or clone is updated to reflect the storage location of the new image while the metadata of all snapshots and clones of that VM or clone is kept unchanged. Clones do not consume any capacity unless data is modified, do not require any upfront space reservation, are created in less than one second, and do not impact the performance of virtual machines. Administrators can take snapshots of clones and build out full snapshot/clone hierarchies with the ability to revert and power up VMs.

A clone is a convenient way to easily duplicate a virtual machine for use in different environments; for example, development and test environments. Clones can be created through MxInsight when viewing VM inventory.

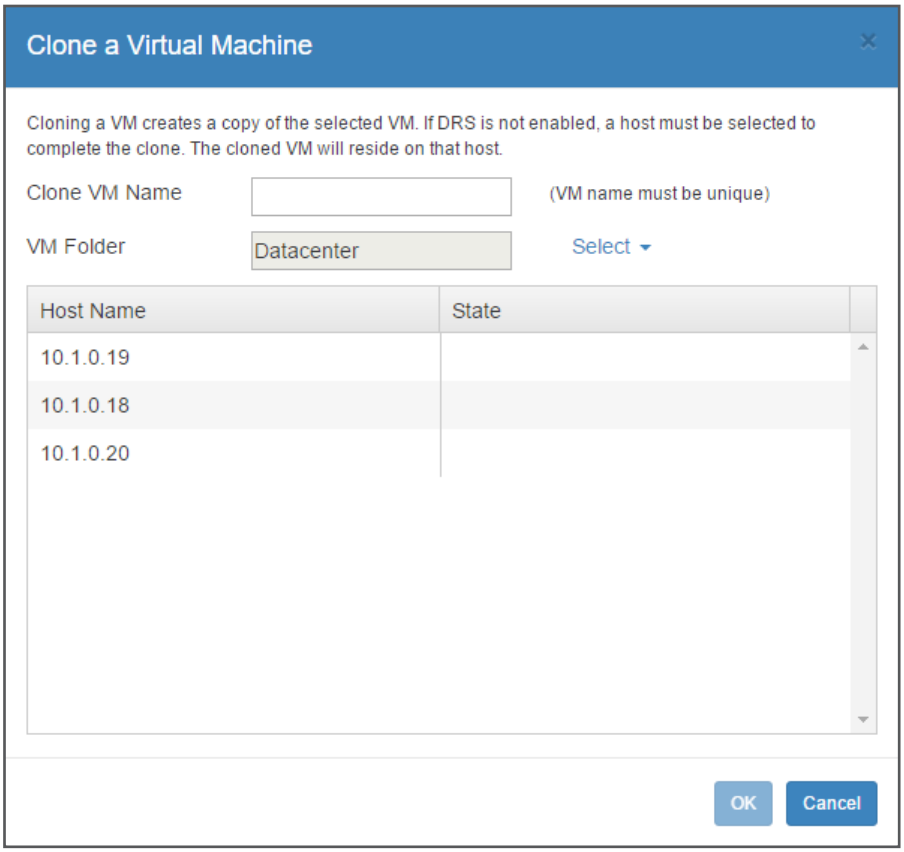


Figure 10: Create a Maxta Clone of a VM

Clone Customization

In addition to creating single clones, MxSP supports automated creation of multiple clones with the option of adding guest customization scripts. Administrators can create multiple clones with unique domain names and other characteristics, including a set scripts to customize the guest operating system when the clone is created. MxSP provides full integration with VMware's guest customization specifications, allowing these specifications to be applied to Maxta clones through MxSP's clone customization wizard.

Figure 11: Create Multiple Clones

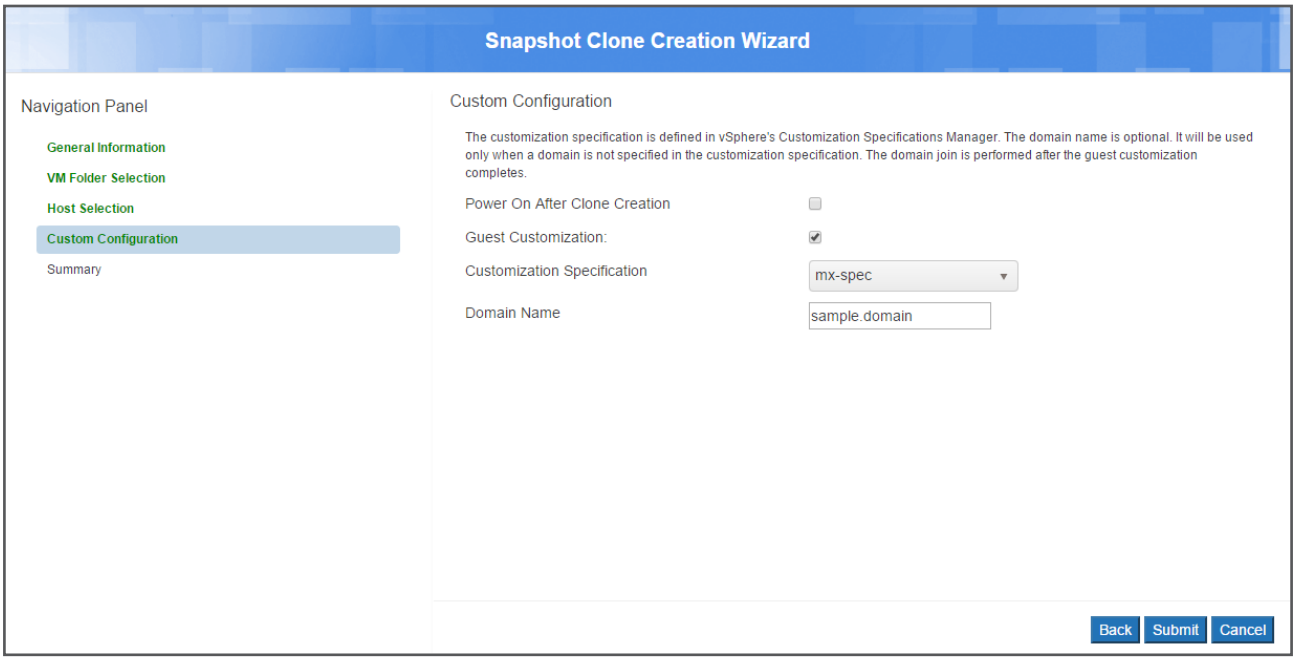


Figure 12: Apply Guest Customization Specification

Data Layout

MxSP optimizes the data layout for the typically random read/write operations of virtual workloads by using a log-based layout approach. The layout is similar to that of log-structured file systems, with highly available metadata maintained for mapping data blocks to their storage locations. MxSP optimizes the mapping of VMs to storage resources. Under normal operation, reads and writes are served by the server hosting the VM. MxSP provides the ability to maximize the probability that a VM's data is read from the node's local cache, eliminating the network round trip latency associated with distributed storage systems.

MFS organizes data in a manner that is designed to maintain an optimal balance across all nodes and disks in the cluster. MFS achieves this by striping and mirroring data across the network while also performing proactive capacity rebalancing. Administrators have the option to set the number of mirrors and the type of data striping in order to meet the specific needs of their cluster's workload. Maxta's Object Manager handles the placement of data based on the given parameters, in addition to prioritizing data rebalance based on capacity.

Object Manager

Object Manager is the primary component responsible for data layout within a Maxta cluster. It determines which disks and nodes should be used to store data when first written to Maxta storage and how to rebalance the data based on capacity usage. Object Manager also handles data re-layout in case of disk or node failure or addition, or re-layout due to a service event such as replacing a node or disk, among other tasks. It immediately places new data on new storage (via node or disk addition) and rebalances old files to the new storage. Finally, Object Manager ensures that data placement abides to all VM or virtual disk storage policies, and restructures data as necessary when these policies change.

Capacity Rebalance

In order to reduce administrative tasks and maintain optimal performance and capacity usage throughout the cluster, MFS has implemented a proactive capacity rebalance across all disks and nodes in the Maxta cluster. One application of this rebalance occurs when a new node or disk is added to the cluster. Rather than only writing new data to this newly added resource, MxSP automatically redistributes data from the other nodes and disks in the cluster onto this new resource. Another use case comes from Maxta's inherent thin provisioning of data. A large file can be thin provisioned onto a node which has minimal capacity, but as the file grows, its data will be written to different nodes and disks across the Maxta cluster. MxSP is thus able to maintain a relatively equal capacity balance across all nodes and disks in the cluster, ensuring that no one resource becomes over-utilized.

MxSP also proactively balances the Maxta cluster for capacity. Rather than waiting for a node to reach a certain capacity limit and then rebalancing data, MxSP continually monitors node and disk capacity usage for deviations from the average of other cluster resources. If there is a large enough variance, MxSP will automatically migrate data in order to restore balance across all nodes and disks in the cluster.

Striping and Mirroring

MxSP both stripes and mirrors (replicates) data in order to create redundancy and maintain availability during hardware failures and when data management policies are modified. MxSP always stripes data, but allows you the choice of striping data horizontally (stripe across all nodes in the cluster) or vertically (stripe within a single node). With either striping method, data is mirrored to retain availability.

MxSP implements horizontal striping by default. The option to horizontally stripe data can be disabled during VM creation, in which case MxSP implements vertical striping. The mirroring policy is also configured during VM creation and can be set to a different number of replicas at any time. Implementing horizontal or vertical striping is dependent on many factors. For example, horizontal striping could be disabled in order to maintain data locality for certain VMs. Maxta enables horizontal striping by default as it provides a greater degree of data redundancy and availability.

Within MFS, a stripe is defined as the smallest amount of data (1 MB by default) that is placed on the disk. A stripe set is a logical entity which represents an aggregation of these stripes. The stripe width (or number of stripe sets) for a file is determined by taking the floor of the number of disks in the cluster divided by the number of replicas for that file. In order to reduce the amount of associated metadata, the maximum number of stripe sets for any file is set to a default value of eight, but this is a configurable parameter. MFS breaks down data into several 1 MB stripes, then based on the previously calculated number of stripe sets, rotates through the 1 MB stripes to form the aggregate stripe sets. Each stripe set is mirrored based on the replica policy to form members, which have an associated Member ID and Storage UUID. Members are written to disks and nodes across the cluster.

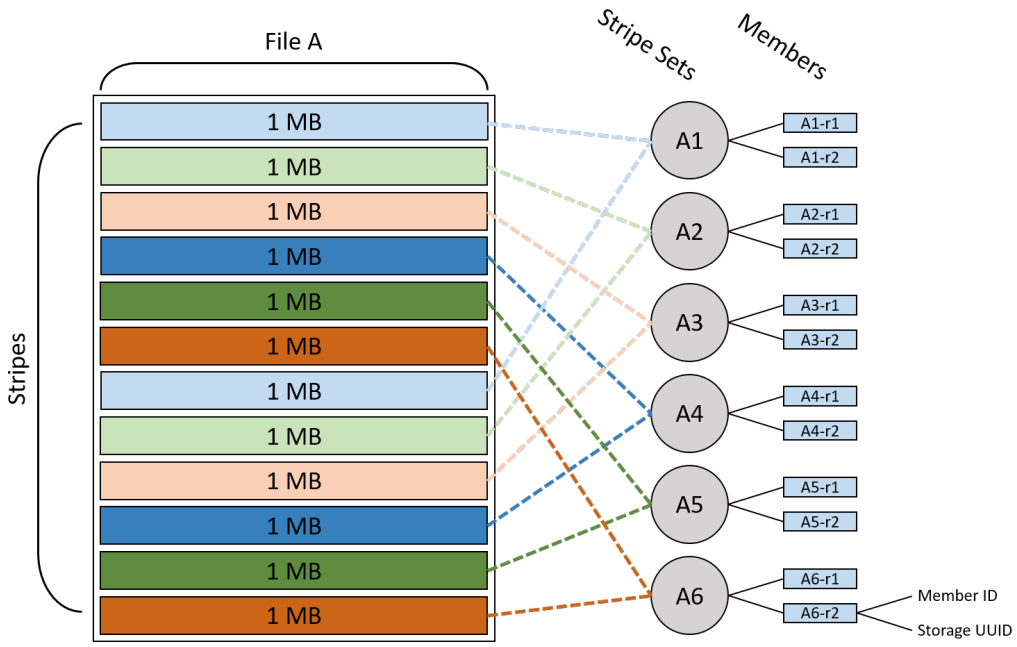


Figure 13: Data Striping

The diagram above shows how a member is composed if there are 12 HDDs in the cluster with a replica policy set to two for a 12 MB file. In this case, there are six stripe sets (A1, A2, A3, A4, A5, A6) since we have 12 disks divided by two replicas. Each stripe set therefore consists of two, 1 MB stripes. The stripes that comprise each stripe set come from non-sequential segments of the data. This is because the stripes are allocated to stripe sets on a rotating basis. For example, in the diagram above, stripe set A1 is composed of stripes 1 and 7. The stripe sets are then mirrored based on the number of replicas defined in the storage policy and are represented as members (A1-r1, A1-r2, A2-r1, A2-r2, A3-r1, A3-r2, etc.). These members (which include the replica copies) are then written to the disks.

With horizontal striping, Maxta stripes data as widely as possible across the nodes in the cluster. Object Manager ensures that a copy of data and its replica are stored on different nodes within the cluster to provide redundancy during node failure. It also stripes data in such a way as to prevent colocation of different members on the same disk. Object Manager prioritizes the placement of data onto storage pools (HDDs) with the most available capacity. Data is first striped and then each stripe set is mirrored on disks and nodes across the cluster.

NOTE: For simplicity, the diagram below shows a scaled-down environment in which each disk only contains one member from one file. Actual deployments will have multiple members from multiple files residing on each disk.

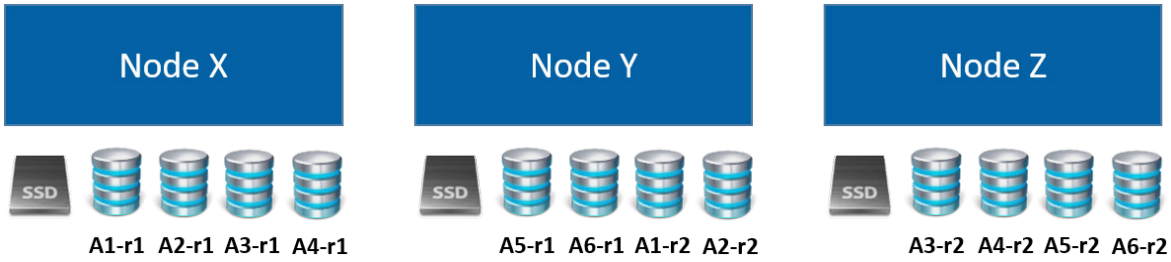


Figure 14: Horizontally Striped Data on Disk

The diagram above illustrates horizontal striping where File A has 12 members (A1-r1 and its replica A1-r2, A2-r1 and its replica A2-r2, etc.) striped across the HDDs in nodes X, Y, and Z. Each member is either an original set of data or that data's replica (meaning that A1-r1 contains the same data as A1-r2). Note that striping and mirroring occurs across nodes.

With vertical striping, data is first mirrored across nodes and then striped across disks within each node. The number of members on a node in a vertically striped environment is equal to the number of available disks on that particular node.

NOTE: For simplicity, the diagram below shows a scaled-down environment in which each disk only contains one member from one file. Actual deployments will have multiple members from multiple files residing on each disk.

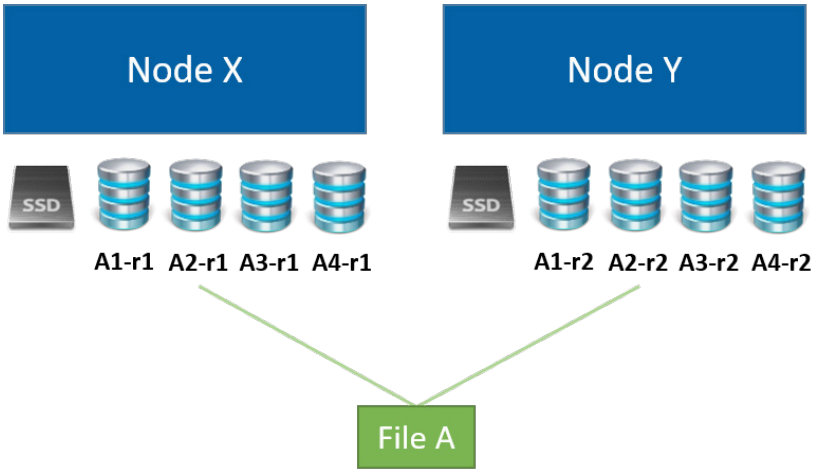


Figure 15: Vertically Striped Data on Disk

In the diagram above, File A is mirrored on HDDs in nodes X and Y, and its members (A1-r1, A2-r1, A3-r1, A4-r1) and respective replicas (A1-r2, A2-r2, A3-r2, A4-r2) are striped on each node.

Object Manager places a few more guidelines on data layout when Maxta Metro Storage Cluster (or Rack Awareness) policies are enabled. One member copy of data is written within a rack (or site/fault domain). The other copy of data (or the replica) is written to a different rack as defined during installation. The horizontal or vertical striping policies described above are maintained for members within each rack. With horizontal striping, this means that each original member (A1-r1, A2-r1, etc.) is striped across the nodes within one rack while each member's replica (A1-r2, A2-r2, etc.) is striped across the nodes within the other rack. For vertical striping, members are striped within one node on one rack and the members' replicas are striped within one node on another rack.

Flash Media Data Layout

MxSP maximizes performance by utilizing flash media in hybrid storage configurations for intent log, read cache, and flash accelerated metadata. Maxta Intent Log (MxIL) is a write cache which consumes a default of 100 GB per node, with its data spread evenly across up to 4 SSD's. MxIL is explained in further detail in the Maxta Intent Log section. Each HDD within a node has an associated Metadev partition of at most 5% of the HDD's capacity. A Metadev partition is written to an SSD and successive Metadev partitions are written across SSD's within the node in a Round Robin (RR) manner. Metadev functionality is explained further in the SSD Caching and Metadev section. Read cache is allocated on an SSD after MxIL and Metadev partitions have been created. For a minimum configuration, there must be at least 100 GB of space available for read cache in order for Maxta installation to succeed (Maxta recommends 300 GB of read cache). When a single node contains multiple SSD's, both MxIL and Read Cache will create equal-sized partitions totaling a minimum of 100 GB each.

Maxta provides optimal failure recovery scenarios via SSD mirroring, which is explained in detail in the Flash Media Failure section. As best practice, Maxta recommends a minimum of two flash media devices (eMLC SSD or PCIe/NVMe flash) per node to support mirroring for failure scenarios.

The tables below explain in further detail how flash media is partitioned based on the number of SSD's within a single Maxta node.

1 SSD

	SSD Mirroring Enabled	*SSD Mirroring Disabled*
MxIL	• Configuration not allowed	• 100+ GB on 1 SSD
Metadev	• Configuration not allowed	• All partitions created on 1 SSD
Read Cache	• Configuration not allowed	• 100+ GB created on 1 SSD

Figure 16: Data Layout with 1 SSD

2 SSD's

	SSD Mirroring Enabled	*SSD Mirroring Disabled*
MxIL	• 100+ GB on 1st SSD • 100+ GB mirrored to 2nd SSD	• 50+ GB on each SSD
Metadev	• Partitions are written in RR-fashion across all SSD's • At the same time, each partition is mirrored to a different SSD	• Partitions are written in RR-fashion across all SSD's
Read Cache	• 100+ GB created across SSD's	• 100+ GB created across SSD's

Figure 17: Data Layout with 2 SSD's

3 SSD's

	SSD Mirroring Enabled	*SSD Mirroring Disabled*
MxIL	• 100+ GB on 1st SSD • 100+ GB mirrored to 2nd SSD • 0 GB on 3rd SSD	• 33+ GB on each SSD
Metadev	• Partitions are written in RR-fashion across all SSD's • At the same time, each partition is mirrored to a different SSD	• Partitions are written in RR-fashion across all SSD's
Read Cache	• 100+ GB created across SSD's	• 100+ GB created across SSD's

Figure 18: Data Layout with 3 SSD's

4 SSD's

	SSD Mirroring Enabled	*SSD Mirroring Disabled*
MxIL	• 50+ GB on 1st SSD • 50+ GB mirrored to 2nd SSD • 50+ GB on 3rd SSD • 50+ GB mirrored to 4th SSD	• 25+ GB on each SSD
Metadev	• Partitions are written in RR-fashion across all SSD's • At the same time, each partition is mirrored to a different SSD	• Partitions are written in RR-fashion across all SSD's
Read Cache	• 100+ GB created across SSD's	• 100+ GB created across SSD's

Figure 19: Data Layout with 4 SSD's

5+ SSD's

	SSD Mirroring Enabled	*SSD Mirroring Disabled*
MxIL	• 50+ GB on 1st SSD • 50+ GB mirrored to 2nd SSD • 50+ GB on 3rd SSD • 50+ GB mirrored to 4th SSD • No data on 5th or more SSD	• 25+ GB on first 4 SSD's • No data on 5th or more SSD
Metadev	• Partitions are written in RR-fashion across all SSD's • At the same time, each partition is mirrored to a different SSD	• Partitions are written in RR-fashion across all SSD's
Read Cache	• 100+ GB created across SSD's	• 100+ GB created across SSD's

Figure 20: Data Layout with 5 or more SSD's

Data Relocation and Reconstruction

There are two forms of data re-layout within MxSP: relocation and reconstruction. The most common data re-layout is data relocation due to capacity rebalance, but relocation can also occur due to node or drive failure. Data relocation simply means that Object Manager is moving an individual member to a new disk or storage pool. Data reconstruction takes place when the storage administrator changes a policy (such as number of replica copies) or a disk or node is added to the cluster. This process involves a complete file re-layout in order to meet the new parameters set by the change in environment.

The workflow described in the following example shows how data relocation works in the case of a disk failure. This example also incorporates the relocation policy as it applies to capacity rebalance.

In the workflow shown in the diagram below:

- There are 12 HDDs and two replica copies.
- The six stripe sets are mirrored, creating a member and its replica (i.e., stripe set A1 has member A1-r1 and replica A1-r2).
- File A has 12 members and replicas (A1-r1, A1-r2, A2-r1, A2-r2, etc.) striped across HDDs in nodes X, Y, and Z.

NOTE: For simplicity, the diagrams below show a scaled-down environment in which each disk only contains one member from one file. Actual deployments will have multiple members from multiple files residing on each disk.

Data relocation for the failed disk occurs as follows:

1) There is a drive failure on Node Z. A new copy of data (A6-r2) is rebuilt from the existing replica (A6-r1). Object Manager places the member onto an available disk that does not contain the member's replica copy. In this case, the data is co-located onto a disk which already contains another member from the same file (A5-r2).

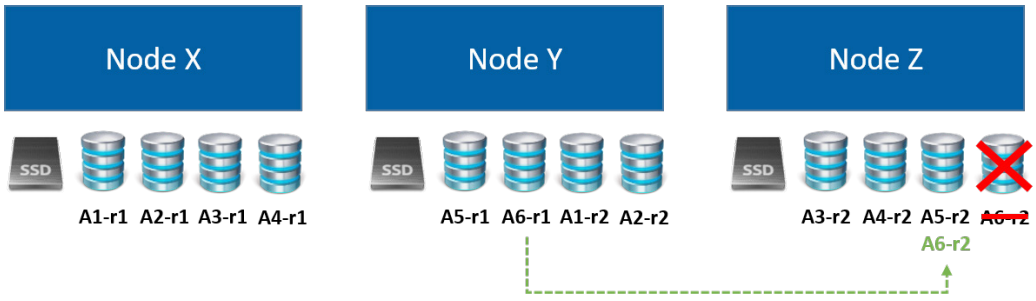


Figure 21: Data Relocation (Step 1)

2) Eventually, a new disk is added to the cluster to replace the failed disk. At this point the workflow becomes the same as that of capacity rebalance. The existing disk on Node Z has two members, the new disk has zero members, and all other disks in the cluster have one member. Object Manager will see the capacity imbalance in the cluster and rebalance the data accordingly. One member from the disk with two members (in this case A6-r2) is moved to the new disk on Node Z.

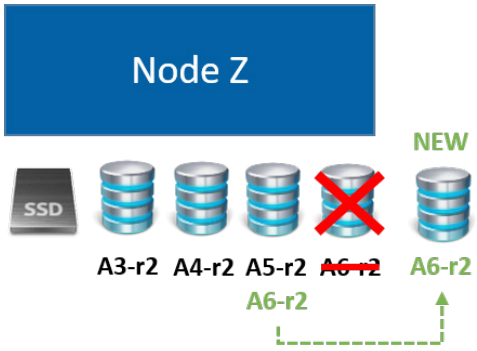


Figure 22: Data Relocation (Step 2)

3) In the final configuration, all disks have exactly one member. The cluster is fully balanced, data is striped as widely as possible, and replica copies are on different nodes to ensure high availability.

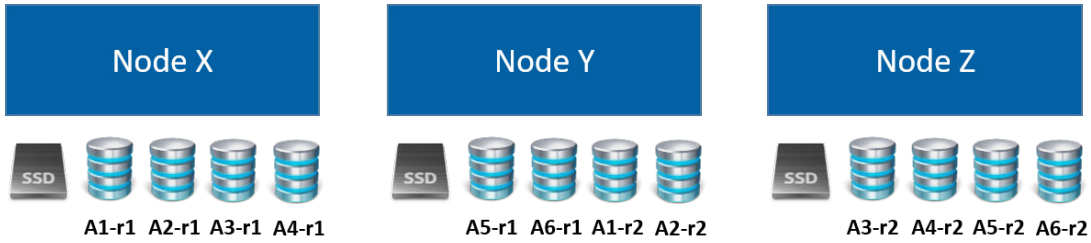


Figure 23: Data Relocation (Step 3)

Object Manager has to re-layout data within MFS when a policy such as number of replicas changes or a disk or node is added to the cluster. The data tree structure, stripe set size, and stripe width are all affected by these changes and therefore need to be adjusted and rebuilt accordingly. In order to maintain zero downtime during this transition, MxSP ensures that all data is constantly available while these changes are being made.

The diagram below shows how Object Manager handles a policy change from two to three replica copies of data.

In the workflow shown in the diagram below:

- There are 12 HDDs and two replica copies originally, meaning there are six stripe sets.
- Each stripe set is mirrored to form a member and its copy (i.e., stripe set A1 has member A1-r1 and copy A1-r2).
- File A has 12 members and replicas (A1-r1, A1-r2, A2-r1, A2-r2, etc.) striped across nodes X, Y, and Z.
- The administrator changes the replica policy from two copies to three copies without adding any additional HDD.

NOTE: For simplicity, the diagrams below show a scaled-down environment in which each disk only contains one member from one file. Actual deployments will have multiple members from multiple files residing on each disk.

Data reconstruction from two replica copies to three replica copies occurs as follows:

1) A new tree is built with the newly calculated number of stripe sets and replica copies. The new tree is marked as stale while data is rebuilt from the old tree to the new tree.

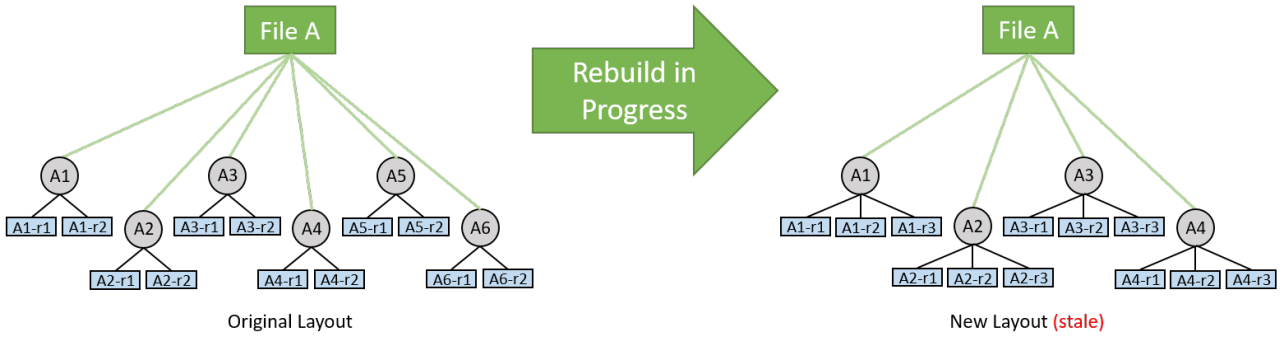


Figure 24: Data Reconstruction (Step 1)

2) When the new tree has finished rebuilding, it is marked up to date and the old tree is deleted.

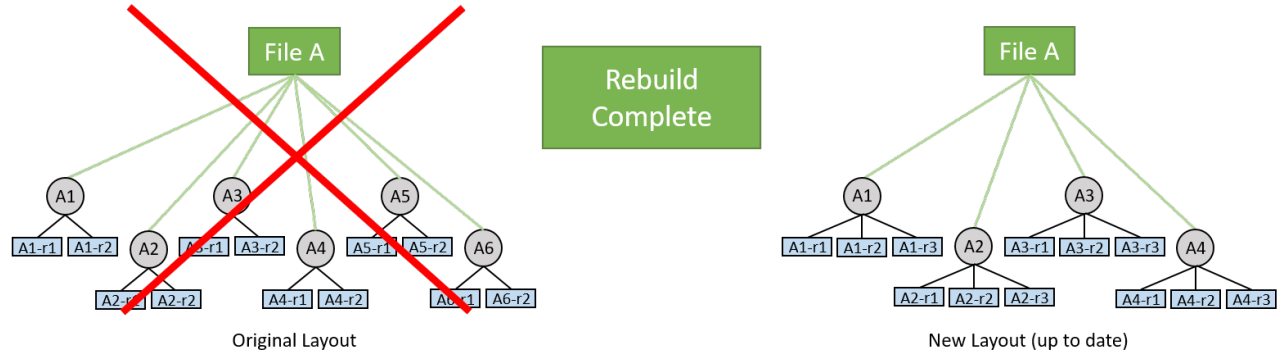


Figure 25: Data Reconstruction (Step 2)

Note that there is a capacity overhead during reconstruction since the entire data tree is copied. This overhead is equal to the size of the virtual disk that is being reconstructed. If Object Manager detects that there is not enough space available in the cluster to support data reconstruction, MxSP will notify the user and data will remain in its existing structure.

The diagram below provides a disk and node-level view of the same data reconstruction process as described above. The new data layout has changed from two replica copies to three replica copies. The administrator has not added HDDs to the environment, causing the stripe width to decrease from six to four. Note that the new data structure which contains an additional replica copy for all data requires an increase in usable capacity equal to the space on disk consumed by one replica copy of the VM.

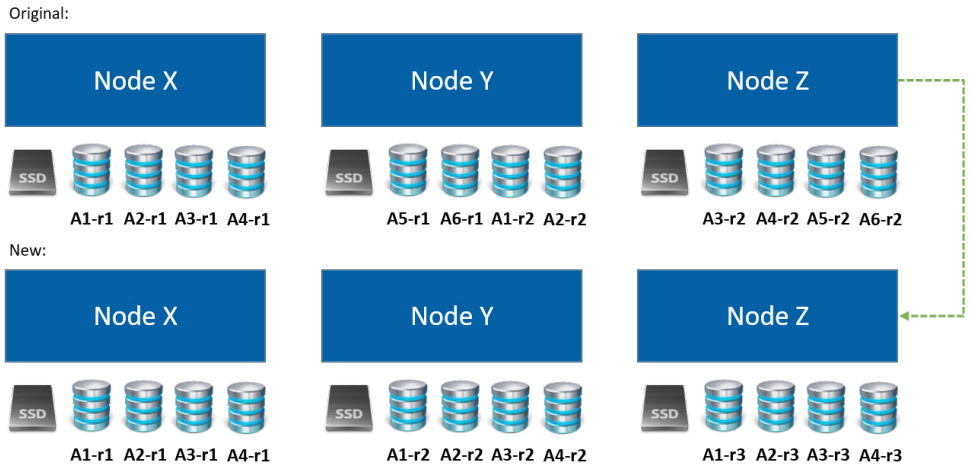


Figure 26: Data Reconstruction on Disk

A similar workflow occurs for other operations that result in a change in stripe width, such as adding a disk or node to the Maxta cluster. In these cases, Object Manager calculates the new stripe width and builds the new tree if there is enough available capacity. Object Manager takes the following pre-emptive steps to minimize any impact on performance and capacity in the cluster:

- Administrators can configure how many files Object Manager will reconstruct at any one time.
- When adding a disk or node, Object Manager will stop reconstruction after determining that the cluster has been reasonably rebalanced rather than forcing reconstruction of all data to meet the new parameters.

Considerations for Optimizing Performance

There are a number of ways to enhance the performance of MxSP. This section provides optimization information for consideration during installation and use of Maxta storage. There are also significant performance improvements which can be realized via Maxta's application defined policies, as discussed in the Policy Management section.

Variable Block Sizes for Maxta Datastore

The block size or page size determines the minimum space allocation unit of a virtual machine on the Maxta datastore. The default page size on the Maxta datastore is 4K. An option is provided during installation to configure the page size of the Maxta datastore and the default page size of virtual machine disks (VMDKs) deployed to the datastore. For example, if the page size is selected as 8K, the minimum space allocation unit for all guest virtual machines would be 8K.

Metadev and SSDs

Maxta Metadev is a class of device in the Maxta file system. Metadev separates metadata from data allowing the metadata to be stored on high performance devices, such as SSDs. This significantly accelerates metadata read and write operations. Metadev is discussed in more detail in the SSD Caching and Metadev section.

Increasing Maxta Virtual Machine Resources

Increasing the amount of memory and CPU resources assigned to a virtual machine can result in performance improvements depending on the workload running on Maxta storage. The default resource reservations of the Maxta

controller VM may prove too minimal to support certain workloads. Issues with insufficient resources can be resolved by increasing the CPU and/or memory reservation through the hypervisor management interface. MxSP seamlessly performs data rebuild operations to ensure that cluster uptime is maintained during this configuration change.

Hardware Components

Being a software-only platform, the underlying hardware consumed by Maxta storage can have some impact on the overall performance experienced by the cluster. They key components to consider for the Maxta hyper-converged solution are storage, networking, and compute. Improved compute resources will enable greater VM density with lower total core count. From a network perspective, Maxta recommends to deploy the MxSP storage network on a 10 GbE network using either a dedicated virtual switch, shared virtual switch with dedicated VLAN, or distributed virtual switch with dedicated port group. Finally, for storage devices, both spindle count and drive type can play a role in overall cluster performance. Higher performance can be achieved in Maxta clusters when there are four or more HDDs per server. There is also a performance gain when using drives capable of higher performance, such as NVMe SSD rather than SATA SSD or 15K RPM SAS HDD in place of 7.2K RPM SATA HDD.

VM Configuration

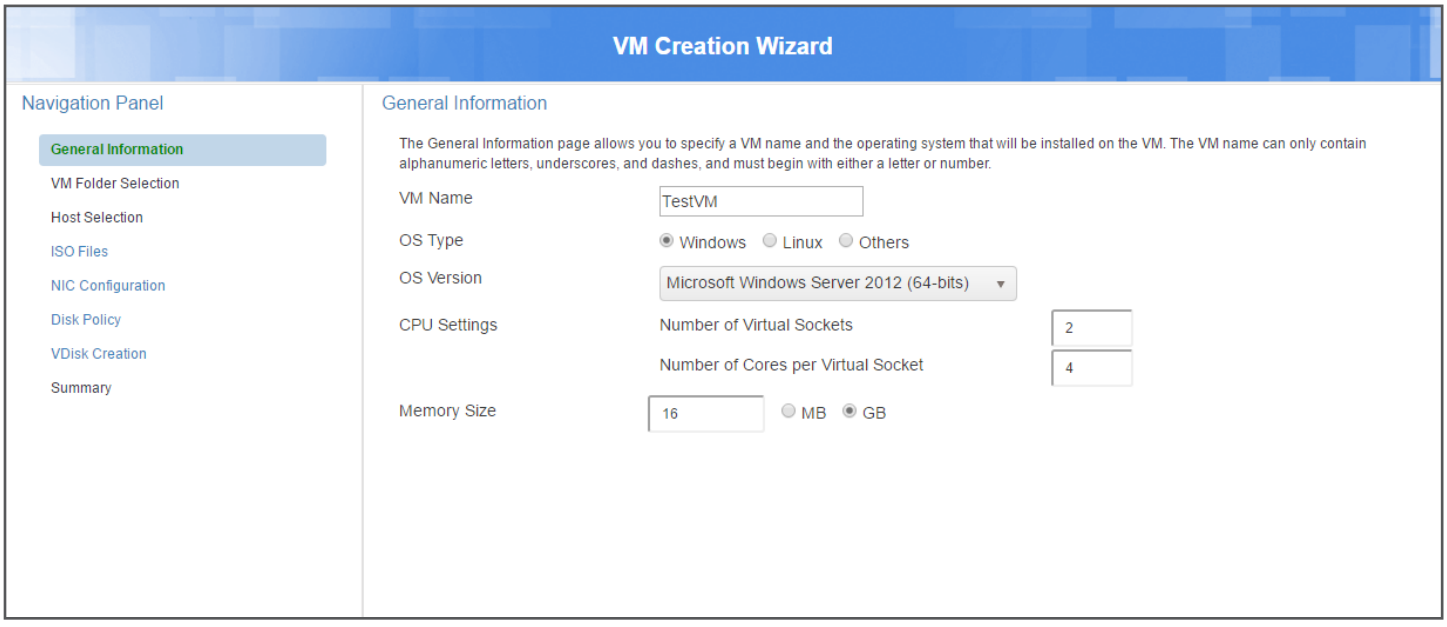
Due to MxSP's striping policies, the configuration of the virtual disks associated with a VM can lead to performance improvements. Multiple virtual disks for each VM will improve the utilization of each physical disk across the Maxta cluster, and thereby improve total cluster performance. Additionally, MxSP is optimized in VMware environments to have data virtual disks which are connected via Paravirtual SCSI controllers.

Policy Management

Maxta's policy management provides further granularity and control over the hyper-converged environment. In keeping with MxSP's VM-centric principles, administrators can set policies on a per-VM or per-vDisk basis. Data layout, capacity optimization, high availability, and other functionality can be set through Maxta's policy management capabilities. MxInsight provides the ability to create policies upon VM creation and modify policies after a VM has been created, whether the VM was created through MxInsight or the hypervisor's centralized management interface. MxSP's policy management also enables automated creation and deletion of snapshots, along with retention schedules. Multi-tenancy is supported via Maxta's integration with user access controls set by the virtual infrastructure management.

VM-level application defined policies can be set when creating a VM through MxInsight or by modifying a VM after it has already been created on Maxta storage. The first few screens of MxInsight's VM Creation Wizard are used to set basic characteristics such as VM name and compute resources.

Figure 27: VM Creation Wizard



Policy on per-VM Basis

When creating a VM through MxInsight, the following policies can be set on a per-VM basis: striping, number of replicas, node affinity, rebuild priority, and metro clustering. All of these policies can be modified after a VM has been created, independent of whether that VM was created through MxInsight or a centralized management tool. However, some of these policies cannot be modified for a VM that is a clone or has dependent clones or snapshots since the new policy would affect the data layout of all parent and/or child objects. The Maxta Management Server configures these properties for all disks and files associated with the VM, and Object Manager applies the properties to the data.

Figure 28: VM-Level Policy

Striping

MxSP allows you to stripe data horizontally (stripe across all nodes in the cluster) or vertically (stripe within a single node). MxSP implements horizontal striping by default (when the striping box is selected). The option to horizontally stripe data can be disabled during VM creation (when the striping box is deselected), in which case MxSP implements vertical striping.

For more information on striping, see the Data Layout section.

Number of Replicas

MxSP supports the ability to synchronously replicate (mirror) data between servers, providing high availability support in the case of drive or node failure. A VM can be configured to have two or more replicas of its data by setting the appropriate value in MxInsight's dropdown menu. This parameter can be set upon creation of the VM, and it can be modified after the VM is already created.

For more information on replicas, see the Data Layout section.

Node Affinity

When node affinity is enabled for a VM, data placement is constrained to a specified set of nodes. A minimum of two nodes must be selected for an affinity group in order to ensure that high availability is maintained. Data for that particular VM will be confined to nodes within the affinity group. Two types of node affinity are available: Soft and Hard. These affinity types determine the behavior of data relocation and reconstruction following a failure event such as node or disk failure.

Under soft node affinity, MxSP's Object Manager will first attempt to rebuild data within the affinity group. If that is not possible due to space constraints or replica exclusions, then the replicas will be relocated to other nodes to ensure availability. If it is a transient node failure, the replica will be moved back to the affinity group when the node eventually returns.

Figure 29: Node Affinity

With hard node affinity, the data is only rebuilt within the specified set of nodes. The replicas will not be rebuilt if there are no available nodes. Therefore, a hard node affinity places strict constraints on data placement, which may lead to situations in which there are no replica copies and data is no longer highly available.

A minimum of three nodes should be included to ensure redundancy when using node affinity. Node affinity policies must be configured with awareness of the available capacity per node.

Rebuild Priority

Rebuild priority can be set on a per-VM basis using a scale of 1 to 10, where 1 indicates highest priority and 10 indicates lowest priority. In the case of failure, VM's with higher rebuild priority will have their replicas rebuilt and relocated before those with lower priority. This targets environments with a few business critical VM's, or applications that must be kept highly available.

Metro Storage Cluster

Maxta's Metro Storage Cluster capabilities can be enabled if Rack Awareness is configured during MxSP installation. A metro storage cluster (or stretched storage cluster) delivers workload mobility, cross-site automated load balancing, and continuous data availability across datacenters.

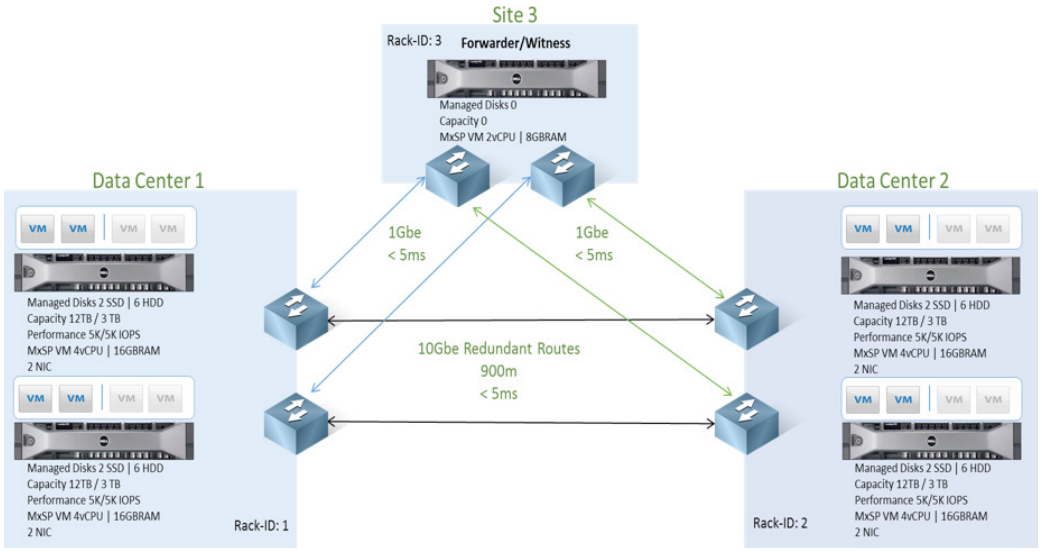


Figure 30: Maxta Metro Storage Cluster

The technology can be adopted to provide continuous availability between racks within a datacenter, across buildings within a campus, or across datacenters spanning different cities. The key requirement to support Maxta Metro Storage Cluster is a maximum latency of 5 milliseconds round-trip time (RTT) across all datacenters for the storage network.

MxSP uses synchronous replication across sites to provide metro storage cluster capabilities. The data I/O path is the same as it would be in normal configuration, wherein we do not acknowledge a write until after receiving an acknowledgment of the write from the remote node. MxSP makes sure that a member and its replica reside on unique nodes in different racks or datacenters as specified via the Rack Awareness installation properties. With this functionality, data availability is preserved even when a drive/server/rack/datacenter fails. MxSP will also intelligently perform reads from the node with lowest latency, therefore read requests will most likely be served from the local rack.

Policy on per-vDisk Basis

During VM creation, the following policies can be set through MxInsight on a per virtual disk basis: page size, compression, and read cache. MxInsight also provides the ability to add a group of multiple vDisks with the same set of policies. All of these policies can be modified after a VM has been created, independent of whether that VM was created through MxInsight or a centralized management tool. When adding a virtual disk to an existing VM, all of the vDisk policies can also be configured. The Maxta Management Server configures these properties for all disks and files associated with the VM, and Object Manager applies the properties for that particular virtual disk.

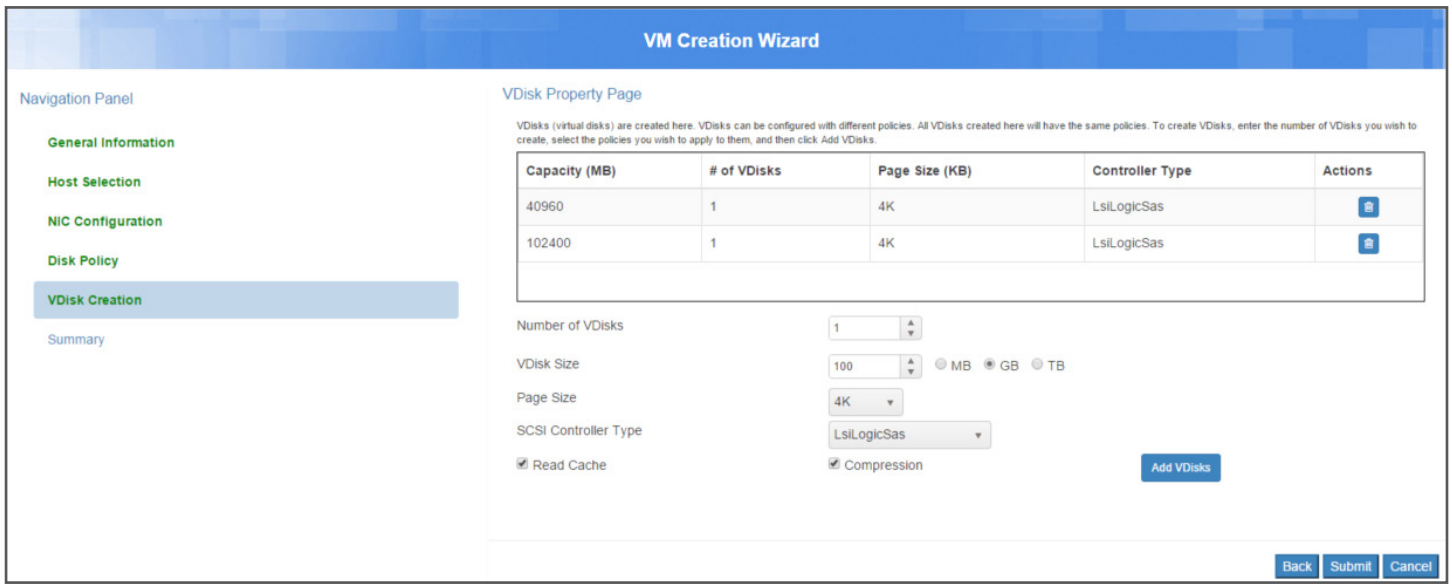


Figure 31: Virtual Disk-Level Policy

Page Size

The page size, or the size of the I/O request made by an application to the storage, generally varies depending on the specific application (email, database, virtual desktop, etc.). MxInsight allows users to set the storage block size on a per-vDisk basis in order to align storage with the application page size. This flexibility enables multiple applications to run on the same Maxta storage with the storage optimized for each individual application. In this way, Maxta environments can support a variety of applications with storage block size optimally configured for each application.

Compression

The compression algorithm utilized by MxSP can be toggled between options for lz4, gzip_opt, gzip_high, or disabled at the time of installation. MxSP performs in-line compression of data and metadata when that data is de-staged from flash to spinning disk. In order to improve the latency of write operations, the data which is stored in the write cache on flash is not compressed. Enabling compression generally improves storage efficiency, but it can come at the cost of higher CPU utilization.

Maxta recommends the use of lz4 in order provide the optimal balance of compression ratio to CPU usage. The gzip_high algorithm will provide the highest compression ratio, but will also result in greatest CPU utilization. The gzip_opt

algorithm will provide better compression ratios than lz4 with a higher CPU usage. Finally, compression can be disabled for uncompressible workloads running on a virtual machine, such as backup or media files.

Read Cache

Maxta's policy management allows for enabling or disabling read cache on a per virtual disk basis. Disabling read cache for a virtual disk will free up space in the SSD's cache, which other applications can use. Disks responsible for certain functions that do not use much, or any, of the read cache, such as a transaction log, could have read cache disabled with minimal to zero performance impact. Additionally, in an environment with thousands of VMs, of which only a few are running read-heavy workloads, it can be beneficial to disable read cache on all but these select VMs. This frees space on the read cache that can then be dedicated to the VMs and disks running read-heavy workloads.

Snapshot Policy

Maxta provides automated data protection via snapshot policies. Snapshot policies create Maxta's zero-copy snapshots for a VM or set of VMs based on a schedule defined by the user. Policies can be configured with a start date and optional end date, a frequency of daily, weekly, or monthly, and a time interval during which snapshots will be taken on a customizable periodic basis. Users can also define how long to retain the snapshots, based either on time period or number of snapshots.

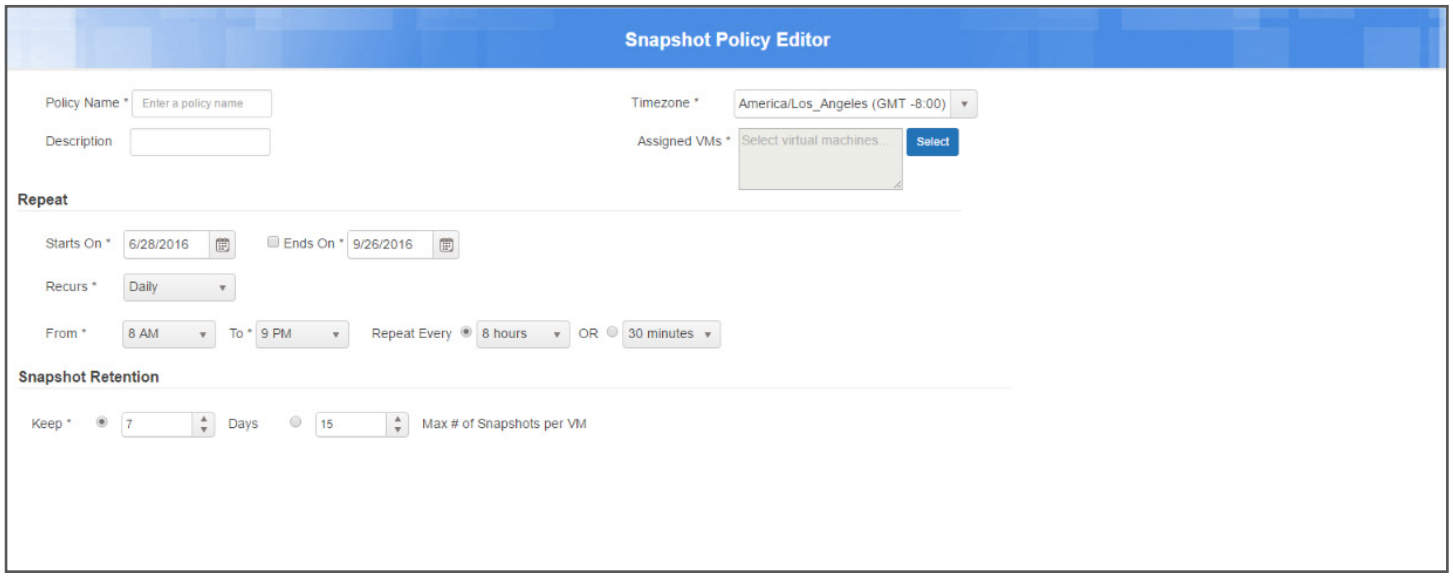


Figure 32: Snapshot Policy Wizard

Maxta's zero-copy snapshots allow for an unlimited number of snapshots to be created and retained without any loss of performance or capacity. The snapshot policy provides data protection by creating a periodic checkpoint of VMs. A failed VM can be recovered from a snapshot which was taken by the policy at a time when the environment was in a working state. In this way, the snapshot policy is a first line of defense against software errors within a VM, providing rapid recovery and data protection in failure scenarios.

Multi-Tenancy

MxSP integrates into existing frameworks for user access control and multi-tenancy. Since Maxta storage is presented to the hypervisor at the virtualization layer, administrators can configure user access control in the fashion they normally would. Maxta will continue to work within this environment with the resource constraints specified under the multi-tenant policies.

Fault Management and Serviceability

Maxta provides reliability and high availability for data, ensuring that all the data in the cluster is protected against error and failure. Strong checksums protect against silent corruption of data that may have been introduced during transmission or storage. Automated recovery ensures that the Maxta cluster does not experience downtime due to any single point of failure.

Data Integrity

All storage platforms use checksums to deliver data integrity of the data stored on the platform. Different storage vendors use different methodology to store and manage checksums. The key goals of data integrity are:

1) To validate the data within a block
2) To validate the block itself

MxSP uses strong checksum algorithms to prevent against silent data corruption in a virtualized infrastructure. Data corruption can be introduced when trying to write a particular data set to disk, but the disk actually returns a different data set. Another form of data corruption comes from misdirected writes, when data is supposed to be written to a particular sector, but is instead written to a different sector of the disk. While not particularly common, these errors do occur, especially with the ever-increasing amount of data being written in modern datacenters.

Maxta protects against these data corruption scenarios by performing checksums at multiple points in the data path. Data integrity is checked whenever the data is stored to a disk, whether that be in MxIL, read cache, or metadev on flash or when data has been de-staged to spinning disk, and anytime the data is sent across the network. Checksums are validated each time data is read. If an error is found, MxSP repairs and recovers the data by patching it with the correct data from an available replica copy.

Traditional storage arrays deliver data integrity by storing the checksum along with the data. This methodology will help validate the data within the block. If the data is corrupted the checksum for the data will not be same as the one stored with the block helping to identify data corruption. This methodology will not assist in validating the block itself. If a wrong block is read but the data within the block is good, the checksum will match the previously stored checksum. Hence this methodology will prevent validating the block itself.

On the other hand, MxSP is able to deliver end-to-end data integrity without any hardware changes to the server configuration. All the underlying blocks of the Maxta storage pool are maintained in a tree structure. To address both the goals “validate the data within a block” and “validate the block itself” the checksum of each block is stored in the parent block and not in the block itself. Every block in the tree contains the checksums for all its children. The root block is handled slightly differently. This is because the root block does not have a parent to store the checksum information. The root block is written (replicated) across various devices in the pool providing the redundancy for the root block.

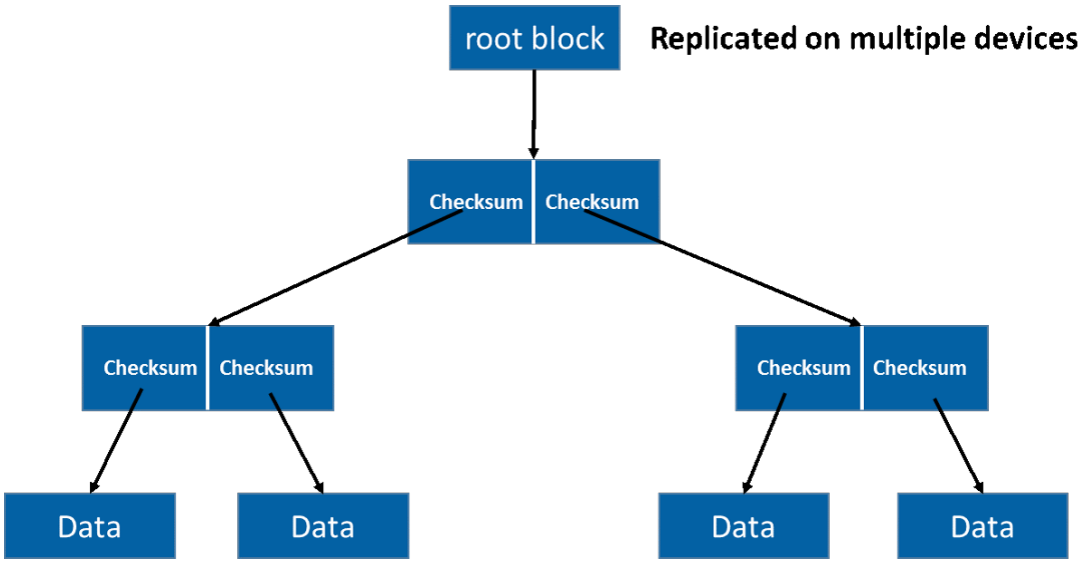


Figure 33: MFS Checksum Tree Structure

When the data and checksum disagree, MFS knows that the checksum can be trusted because the checksum itself is part of some other block that’s one level higher in the tree, and that block has already been validated. With this approach the entire pool validates itself, and MxSP is able to detect and correct silent data corruption.

Types of Failures

MxSP has been developed with a focus on quickly and easily recovering from failures with zero data loss and zero downtime. MxSP has been designed to avoid having any single point of failure in the cluster and to be able to quickly and automatically recover from any failure. The storage platform is able to withstand failures to spinning disk, flash media, servers, sites, racks, and Maxta services without suffering any downtime or loss of data. MxSP's resiliency covers both transient and permanent failures. Additionally, Maxta is able to leverage the hypervisor's built-in high availability protection, such as using NIC-teaming to protect against network failure.

Maxta's policy management features play an important role in high availability scenarios. The copies of data maintained in replicas enable rapid recovery from failures, and additional replicas ensure that the datacenter can withstand multiple failures without seeing any downtime. For example, a VM deployed on the Maxta cluster with three replicas of data will persist through two HDD failures without data loss or downtime.

See the Data Layout section for more information on the how data is relocated or reconstructed after a failure.

Replica Resync

The primary concept to understand when it comes to recovering from failure scenarios in a Maxta environment is that of replica resync. As mentioned throughout this document, MxSP creates a minimum of two copies of all data in order to maintain replicas across the nodes in the Maxta cluster. If at any point one of the copies of data becomes unavailable, Maxta storage will still be able to use the other existing replica copy. Maintaining the replica copy ensures that there is no downtime or data loss in failure scenarios.

When a replica copy of data is lost, MxSP will mark that copy of data on the particular disk or node as stale, and MxSP will initiate a replica resync once it has determined that the replica copy is permanently lost. Maxta also has the ability to perform incremental rebuilds when there is a transient failure. In this case, with replica data only unavailable over a temporary time period, MxSP will perform a resync of just the portions of data which were not written to the replica copy rather than rebuilding the entire data set. This intelligence greatly reduces network traffic and enables quicker failure recovery.

Replica resync and rebuild of data across nodes is fundamental to the way in which Maxta recovers from various types of failure scenarios. MxSP treats each individual disk as an object contributing to the storage pool, and this level of granularity allows for most failures to be handled using similar workflows.

Site/Rack Failure

Maxta uses the concept of a metro storage cluster to protect datacenters against failure of an entire site. Maxta Metro Storage Cluster is enabled by default when Rack Awareness is configured during installation. With Rack Awareness, Rack IDs are assigned to individual servers in order to specify which servers are located on which racks or sites. This information provides MxSP with the details necessary to place each replica copy of data at a different location, ensuring that failure at any single rack or site will not cause any downtime or loss of data.

Node Failure

Node failures can be broken down into two categories: transient and permanent. In the case of a transient failure, MxSP leverages checkpoints to perform an incremental rebuild of replica data. If there is a permanent failure, MxSP will use the existing replica copy to rebuild the data on another node. In both cases, a replica resync is performed to return the cluster to a state of high availability and resiliency.

Disk Failure

Since each drive is treated as a unique object, a drive in the Maxta storage pool can be treated similarly to a node during a failure scenario. The main difference between node and drive failures is that there is no concept of a transient failure when it comes to drives. Anytime a drive fails in the Maxta cluster, the data from that drive is rebuilt to other disks in the cluster immediately.

Maxta Storage does not have any concept of pairing drives for replication, giving the benefit of allowing all available devices to be treated as spares during a rebuild. This means that the entire set of data on a particular drive does not

need to be fully copied to another drive; instead, MxSP is able to break up the data and distribute it to multiple drives across the cluster. This technique avoids choke points wherein there would not be enough capacity on a single disk to support a full rebuild of all the data from the failed disk.

Flash Media Failure

Similar to node and spinning disk, flash media failures do not result in any downtime or data loss. The Maxta installer provides the option to mirror data stored in flash. If mirroring is enabled, then a failure will have no impact – the failed device's mirror will be used and once the failed device is replaced, the environment is restored to its highly available state. However, flash media failure will result in a full node rebuild if mirroring is disabled. A full explanation of flash media failure scenarios can be more easily described by breaking it down into the three components for which MxSP uses flash: read cache, Metadev, and intent log.

Failure as it pertains to read cache is simple – there is no loss of data. The read cache maintains a cache of data in flash so that fewer reads come from the slower spinning disk. When we lose this partition of data on the flash device, there will be some level of performance degradation due to less total capacity being available for the read cache. However, there is no data loss since the flash media only acts as a cache of the data that is already stored on the spinning disk.

If flash media mirroring is enabled, failure of a single flash device will not result in loss of data for the Metadev partition. In this case, metadata is mirrored across two flash devices on a single node; therefore, once the failed device is replaced, the device will resync with its mirror and the cluster will return to its highly available state. When mirroring is disabled, MxSP will perform a full node rebuild.

MxIL data is also mirrored across flash media within a single node when flash media mirroring is enabled. Similar to the Metadev partition, there is no data loss and the MxIL partition will resync with its mirror when the failed device is replaced. In non-mirrored configurations, it is important to remember that MxSP creates a minimum of two copies of all data. Therefore, the default I/O path sends data to at least two nodes on every write, and MxSP can recover from flash media failure by rebuilding data from the intent log of the other node. MxSP only needs to perform an incremental rebuild of the data which has not yet been de-staged from flash to spinning disk.

Service Failure

Since MxSP runs in the user space (as a VM in VMware environments and as a process in KVM environments), there are no kernel panics when any associated services fail. Instead, users can simply restart the service and resume using the product. Maxta has built some additional functionality to eliminate downtime in the event that some of these services fail for any reason.

NFS IP Failover

The Maxta management server VM deployed on every node in the Maxta cluster has three private network interfaces: NFS, Peer, and Standby. Maxta Storage services communicate across nodes using the Peer interface and associated Peer IP address, which is a fixed address that is always owned and activated by the node itself. The Standby interface of a node is usually inactive and will not have an IP address assigned under normal operating conditions. Each hypervisor in the Maxta cluster connects to the NFS server using the NFS interface and associated NFS IP address. The NFS IP address is a floating IP address which is owned by the node and activated subject to winning an NFS IP leader election. When a node comes up in the Maxta cluster, it activates the Peer interface, leaves the Standby interface idle, and runs an NFS IP leader election to acquire its NFS IP – the node will always win the election to acquire its own NFS IP.

If a service on the management server goes down, or the node becomes unresponsive, all nodes with an idle Standby interface will run an NFS IP leader election to take over the NFS IP of the failed node. The winner of this election activates the failed-over NFS IP address on its standby interface. Throughout this process, there is no downtime or data loss, and the administrator will continue to have access to Maxta storage from all nodes in the cluster. When the previously failed node returns to the cluster, it will run an election to re-acquire its NFS IP address (which, as previously stated, a node will always win the election to acquire its own NFS IP address), and the previous winner will return its Standby interface to an idle state. In this final stage, the cluster has returned to its normal state, and storage traffic is once again directed locally on the node.

Auto Migration

In order to maintain quorum and fault tolerance, MxSP guarantees that its unified namespace service is always running on three nodes within the Maxta cluster. The auto migration of this service is similar to the NFS IP failover scenario, but does have a few key differences which are highlighted below.

Upon deployment of the Maxta cluster, a leadership election is performed to select a master node. This master is responsible for re-assigning the namespace service to another node in case of failure. If the master node itself fails, the leadership election is run against the remaining nodes in the cluster to select a new master node. In a rack-aware cluster, auto migration will ensure that the namespace service runs on three distinct sites as long as nodes are available in that configuration. When a node within a particular rack fails, auto migration will start the namespace service on a node within that same rack if possible.

In addition to providing fault tolerance in a reactive manner, auto migration also has a proactive balancing functionality. A balanced cluster in this sense means that the unified namespace service is running across three distinct sites when possible. Proactively balancing the cluster would occur when there is one rack with multiple nodes running the unified namespace service and another rack which has no nodes running the service. In this scenario, MxSP will automatically migrate the namespace service to the other rack in order to balance the cluster. This situation could occur when a node is added to a new rack in the cluster or when the nodes in a site recover after failure (during which time the namespace service would have been migrated to other racks, potentially creating an uneven balance).

Adding Hardware

MxSP supports the ability to scale up and scale out in order to meet the exact the needs of a growing datacenter. HDD, SSD, and node addition procedures can be performed via MxInsight and the Maxta CLI in order to match cluster size to environmental requirements.

Adding Disks

New HDDs and SSDs can be added to increase the capacity of the Maxta storage pool or maintain the HDD to SSD capacity ratio for Metadev-enabled clusters. A single disk or multiple disks can be added to an individual server or across all servers to increase available capacity. After additional capacity has been added to the cluster, MxSP will perform data reconstruct actions in order to maintain even distribution of workload across all nodes. Disk addition can be performed via the Maxta CLI using the mxServices command to register the disk with MxSP and add it as a storage pool to the Maxta cluster.

First, list the disks available on each server:

```
console:~$ mxServices -L
NODE: 01
 - DISK: 01
  - /dev/disk/by-id/scsi-36000c29765eaf96add98621e0815b04a
  - 414 GB
  - State: ONLINE
...
 - UNREGISTERED DISK: 00
  - /dev/disk/by-id/scsi-1ATA_ST9750420AS_6WS03N78
  - 698 GB
```

Then add the disk to the Maxta node:

```
console:~$ mxServices -n 01 -D 00 -o add
WARNING: Any data left on the disk may be wiped. Confirm? (Y/N): y
Starting disk addition process...
Operation ID: op_001
Please use op-status and node id to monitor the status of the disk addition.
```

Finally, monitor the progress of the disk addition operation:

```
console:~$ op_status -n 01 -i op_001
Status: 30% 0% |=============>----------------------------------| 100%
```

When the disk has been added to the Maxta storage pool, it will show up as a registered disk on the node:

```
console:~$ mxServices –L
NODE: 01
  - DISK: 01
    - /dev/disk/by-id/scsi-36000c29765eaf96add98621e-
0815b04a
    - 414 GB
    - State: ONLINE
…
 - DISK: 05
    - /dev/disk/by-id/scsi-1ATA_ST9750420AS_6WS03N78
    - 698 GB
    - State: ONLINE
```

Adding Nodes

New nodes can be added to expand the cluster capacity and improve cluster performance. Existing nodes can also be replaced or upgraded as needed within the Maxta cluster. Once a node has been added to the virtual infrastructure's cluster, including all necessary networking and storage devices, the node can be added to the Maxta cluster via MxInsight.

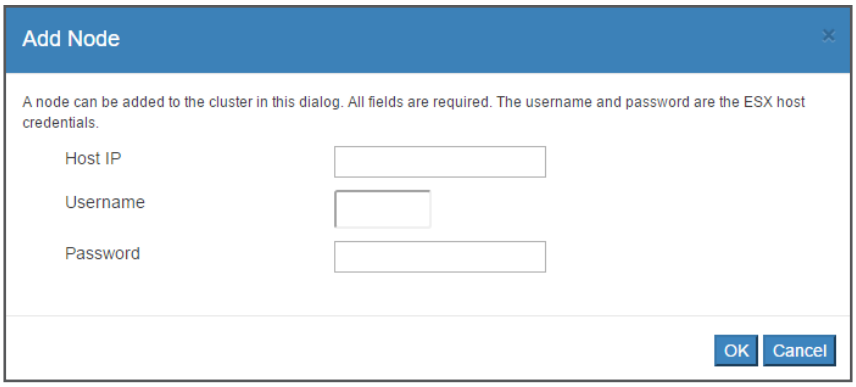


Figure 34: Add Node in MxInsight

Replacing Hardware

Servers and disks within a cluster may need to be replaced periodically due to hardware failure or in order to increase cluster capacity or performance. Workflows for replacing hardware are similar to that of adding new hardware to Maxta clusters. The table below provides more information regarding data resync operations that occur based on the specific operation performed in different configurations of Maxta clusters.

Object	Service Action	Result
HDD	Add a new HDD to increase storage capacity	• Data is reconstructed across the cluster • Once the new disk is registered, MxSP automatically rebalances the data in the cluster to ensure the widest possible stripe across the nodes.
	Replace an existing HDD with a higher capacity HDD	• MxSP relocates data from the failed disk onto a good disk in the cluster. • Once the new disk is registered, MxSP automatically rebalances the data in the cluster to ensure the widest possible stripe across the nodes.
• **HDD** • **SSD (mirrored)**	Replace a failed HDD/SSD with a new HDD/SSD	• MxSP relocates data from the failed disk onto a good disk in the cluster. • Once the new disk is registered, MxSP automatically rebalances the data in the cluster to ensure the widest possible stripe across the nodes.
SSD (mirrored)	Replace an existing SSD with a higher capacity SSD	• Data contained on the removed SSD exists on the other, mirrored, SSD in the node. • Once the new disk is registered, MxSP automatically rebuilds MxIL and Metadev on that disk.
	Add a new SSD to increase SSD capacity	Once the new disk is registered, MxSP automatically adds its capacity to the available pool for Metadev and read cache.
SSD (non-mirrored)	Replace a failed SSD with a new SSD	• When the node is marked as failed, MxSP removes the node on which the failed disk resides from the Maxta cluster. • The removed node's data is reconstructed across the cluster. • Once the SSD replacement has been physically installed, the node must be added back to the cluster. • Once the node has been added back to the cluster, MxSP automatically rebalances the data in the cluster.
Node	Add a new node to the cluster	When the new node has been successfully added, MxSP rebalances data across the cluster
	Node maintenance – node remains in MISSING state	If the node remains in the MISSING state, data is marked as stale and then incrementally resynced after the node comes back online.

Object	Service Action	Result
	Node maintenance – node has entered FAILED state	• MxSP relocates the data from the failed node to a good node in the cluster. Data is relocated to ensure that I/O is not disrupted and high availability is maintained. • Although there is no issue with the node hardware, because the node was marked as FAILED, the node must be removed and then re-added to the cluster. • When the new node has been successfully added, MxSP rebalances data across the cluster.
	Replace a node – node remains in MISSING state	If the node is replaced while in the MISSING state, data is marked as stale and then resynced after the new node is added to the cluster.
	Replace a node – node has entered FAILED state	• MxSP relocates the data from the failed node to a good node in the cluster. Data is relocated to ensure that I/O is not disrupted and high availability is maintained. • When the new node has been successfully added, MxSP rebalances data across the cluster.

Management

Maxta Management Service

The Maxta management service provides administrators with a means of accessing MxSP services via the public network interface. This service only runs on one Maxta controller VM in the Maxta cluster at any point in time. The VM will own the IP address specified in the MxSP installer and become the host for the Maxta management service.

The management service is a gateway to MxSP and a way for users to establish SSH connections to the other Maxta controller VMs. Administrators can interact with the Maxta Command Line Interface (CLI) and the same REST APIs that MxInsight uses to interface with MxSP. Additionally, MxCloudConnect and other services are initiated by the management service on the same Maxta Controller VM on which the management service is running.

Maxta management service is stateless – its state is contained in the unified namespace – and can therefore ensure high availability. When the controller VM hosting the management service fails, a leadership election is performed in the Maxta cluster and a new node will take over the management role. The Maxta controller VM on that node will activate its public interface and use the same IP address that was assigned for management during installation. Services which run on the Maxta Controller VM, such as MxCloudConnect, inherit the high availability of the management service – when the management service is brought up on a new node, it will also start up the associated services on that node.

Maxta Command Line Interface and REST APIs

The Maxta Management Server provides an access point into the Maxta CLI and REST APIs. Administrators can leverage the CLI, Web Services API (WSAPI), or API client to facilitate cluster management tasks.

Maxta CLI

Administrators can log in to the management server as the console user and directly interface with MxSP using the Maxta CLI. One example of an operation that can be performed through the CLI is disk addition:

```
console:~$ mxServices –n 01 -L
NODE: 01
 - DISK: 01
  - /dev/disk/by-id/scsi-36000c2971efee85090d-
85a27017be756
  - 414 GB
  - State: ONLINE
 - UNREGISTERED DISK: 00
  - /dev/disk/by-id/scsi-1ATA_ST9750420AS_6WS03N78
  - 698 GB
console:~$ mxServices -n 01 -D 00 -o add
```

Another operation that can be performed via Maxta CLI is to change the MxSP management IP address. The command below changes the management IP address from 10.3.2.50 to 10.3.2.51:

```
console:~$ mxConfig
************************************************
Enter vCenter IP (or FQDN) to move nodes.
************************************************

************************************************
 Current Settings:
------------------------------------------------
    Management IP Address = 10.3.2.50
            Primary DNS = 10.2.1.2
          Default Gateway = 10.0.0.254
            Subnet Mask = 255.240.0.0
************************************************

usage: com.maxta.mxutilities.maintenance.MxConfig
 -c,--cluster <cluster>        cluster to move Maxta nodes.
 -d,--dns <dns>                (optional) primary DNS.
 -e,--extension                (optional) register VC extension only.
 -g,--gateway <gateway>        (optional) default gateway.
 -m,--mgmtip <mgmtip>          IP (or FQDN) of management server.
 -p,--password <password>      password of vCenter instance. You will be
                               prompted to enter if it is not specified.
 -s,--subnetmask <subnetmask>  (optional) subnet mask.
 -u,--username <username>      username of vCenter instance.
 -v,--vcip <vcip>              IP (or FQDN) of vCenter instance.
console:~$ mxConfig -v 10.15.10.250 -u root -p password -c mxcluster -m 10.3.2.51
```

Maxta API

Maxta's REST APIs are exposed via the Maxta Management Server, providing GET and POST functions for all features of MxSP. Maxta's Web Services API (WSAPI) consists of an application server and a definition of the WSAPI operations, inputs, and outputs. Maxta's WSAPI adheres to respresentational state transfer (REST) architectural constraints. The WSAPI provides more flexibility in handling storage tasks than Maxta's CLI or MxInsight and can be used to automate various cluster and virtual machine management tasks. Clients communicate via HTTP requests and data structures represented with JavaScript Object Notation (JSON). Maxta's API client provides the same virtual machine management as the WSAPI, except through the command line.

The following HTTP methods are supported by the Maxta WSAPI:

Method	Description
GET	Retrieves information identified by the request URI.
POST	Requests that a creation action be performed as specified by the URI.
PUT	Requests that a modification action be performed as specified by the URI.

The requests in the table below are available for use with the Maxta WSAPI, with the supported URI format of http://<management server>/<request>.

Request	Method	Description
/api/v3/base/cluster	GET	Get cluster information
/api/v3/hosts	GET	Get host information
/api/v3/virtual-machines/default-policy	GET	Get cluster policy information
/api/v3/base/folders	GET	Get folder information
/data/vcenter-window-version.json	GET	Get Windows OS version
/data/vcenter-linux-version.json	GET	Get Linux OS version
/data/vcenter-other-version.json	GET	Get other OS versions
/api/v3/license	GET	Get Maxta software license information
/api/v3/version	GET	Get MxInsight version information
/api/v3/base/stores/host-1941	GET	Get ISO store information
/api/v3/base/stores/contents/<ISO folder name>	GET	Get child folder information
/api/v3/base/stores/contents/<ISO folder>?path=<ISO path>	GET	Get ISO image information
/api/v3/virtual-machines	POST	Create a virtual machine
/api/v3/base/datastore/virtual-Machines	GET	Get a list of current virtual machines
/api/v3/virtual-machines/<VM ID>	GET	Get a virtual machine's policy
/api/v3/virtual-machines/<VM ID>	PUT	Modify a virtual machine's policy
/api/v3/tasks	POST	• Power a virtual machine on or off • Delete a virtual machine • Create a snapshot • Create a clone • Check task status

All operations that can be performed through MxInsight are also available as Maxta APIs. As an example, an administrator can set and get license information for a Maxta cluster:

```
Request: - check license status
Request Url: http://10.15.28.43/api/v3/license?sta-
tus
Request Method: GET
Status Code: 200
Params: {}
Response:
{
"data": {
        1. "status": 1,
        2. "isExpired": false,
        3. "expirationDate": "2015-03-02T00:00:00Z"
       }
}
```

```
Request: - import license
Request Url: http://10.15.28.43/api/v3/license
Request Method: POST
Status Code: 200
Params: {}
Body:
{encodedLicenseString: “-----BEGIN PGP MESSAGE-----<License Key>-----END PGP MESSAGE-----”}
Response:
Status Code: 200
Date: Tue, 20 Jan 2015 20:36:33 GMT
Server: Apache-Coyote/1.1
Transfer-Encoding: chunked
Content-Type: application/json;charset=UTF-8
{
“data”: {
“encodedLicenseString”: null,
“installationTime”: null,
“licenseFeatures”: null
}
}
```

For further documentation of the Maxta CLI, please see the Maxta User Guide.

```
Request: - get license content
Request Url: http://10.15.28.43/api/v3/license
Request Method: GET
Status Code: 200
Params: {}
Response:
{
“data”: {
        1.      “-----BEGIN PGP MESSAGE-----<License Key>-----END PGP MES-
SAGE-----”,
        2.      “installationTime”: “1970-01-01T00:00:00Z”,
        3.      “licenseFeatures”: {
        4.          “defaults”: {}
        5.      }
 }
}
```

Maxta Graphical User Interface – MxInsight

Maxta’s management console, MxInsight, is a responsive web application which can be accessed in two ways: a single pane of glass integrated into the hypervisor’s virtualization user interface or its own webpage. MxInsight provides visibility at VM-level granularity, with no need to manage LUNs or other storage constructs.

The management console has been designed to provide administrators with more visibility into the status and performance of the Maxta cluster. The dashboard provides a quick overview of all relevant data for that cluster, such as health, capacity usage, performance statistics, and inventory information. There is also a quick access menu for performing common tasks such as creating a VM.

Digging deeper into MxInsight reveals historical and real-time statistics related to capacity, latency, IOPS, and throughput. There are also statistics for each node, virtual machine, and disk to give administrators further insight into the performance of their cluster. MxInsight provides disk information on a per-node basis, allowing administrators to view a breakdown of storage pool capacity usage for each disk in the Maxta cluster, including SSDs and flash media.

Administrators can monitor events, alarms, and recent tasks through MxInsight, as well as view the status of ongoing tasks. MxInsight can be configured to trigger alerts at different thresholds for each indicator, including SSD wear. There are also options to configure language preferences and email notifications via SMTP.

Figure 35: MxInsight Dashboard

MxInsight streamlines the workflow of service actions such as adding and removing disks and hosts from the Maxta cluster. Maxta's user interface allows administrators to create and manage VMs with the standard set of features available in the hypervisor manager, as well as policy management functionality unique to MxSP. Individual VM and virtual disk policies can be configured during VM creation and modified after VM creation, whether that VM was created through MxInsight or the hypervisor management console. Administrators can set policies for creation and retention of snapshots at specified time intervals, and they can automate the creation of multiple clones with customization specifications.

MxSP Statistics

MxSP uses a statistics database to maintain historical data points for various metrics as they relate to the Maxta storage cluster. These metrics are collected at cluster, node, storage pool, virtual machine, and virtual disk level in order to provide administrators with higher levels of granularity into the performance of their storage infrastructure. The parameters for how frequently to gather data and how long to store the data can be configured by the administrator. These statistics are displayed in MxInsight and are available through Maxta's REST APIs.

The following metrics are stored in Maxta's statistics database:

- Used and free **capacity** on cluster, node, and storage pool levels
- Read and write **latency** on cluster, node, storage pool, virtual machine, and virtual disk levels
- Read and write **IOPS** on cluster and node levels
- Read and write **throughput** on cluster and node levels
- Percentage of **cache hits** at VM and virtual disk level, broken down into MxIL, in-memory, read cache, and misses

Cluster Statistics

Detailed real-time information about the cluster's capacity and performance (latency, IOPs, and throughput) is displayed in the Cluster Statistics page within MxInsight. Each set of statistics also includes the date range and polling interval from which the data was gathered. The graphs are refreshed each minute, with the far right of the graphs showing the most current data. Additionally, historical charts are also available by expanding any of the graphs. The cluster-wide performance metrics can be used to determine if there are trends with regard to large IOPs dips during read and write operations due to issues with your hardware, software, and/or network setup.

The statistics displayed in MxInsight are:

- Cluster Capacity Details - displays a view of the cluster's capacity usage and availability in GB.
- Cluster Latency Details- provides a view of the cluster's read and write latency. This performance metric can be used to determine if there are any trends with regard to high latency spikes during read and write operations due to issues with hardware, software, and/or network setup.
- IOPs, Cluster IOPs Details - shows a view of the number of read and write operations in your cluster.
- Throughput, Cluster Throughput Details - displays a view of the throughput (in MB per second) for reads and writes in your cluster. As with the latency and IOPs metrics, use the throughput performance metric to determine if there are any throughput anomalies during read and write operations due to issues with your hardware, software, and/or network setup.

Figure 36: Cluster Statistics

Node Statistics

MxInsight presents a menu option to view statistics at a per-node granularity. The charts which are displayed by default represent real-time node-level statistics. Historical statistics can be viewed through a drop-down menu accessible above the graphs.

The statistics displayed on a per-node basis are:

- Available Capacity - displays a view of the node's capacity usage and availability in GB. The available capacity will reflect the fact that MxSP automatically balances data across the nodes in the cluster.
- Latency - provides a view of the node's latency performance.
- IOPs - shows a view of the number of read and write operations.
- Throughput - displays a summary and a graphical view of the throughput (in MBs per second).

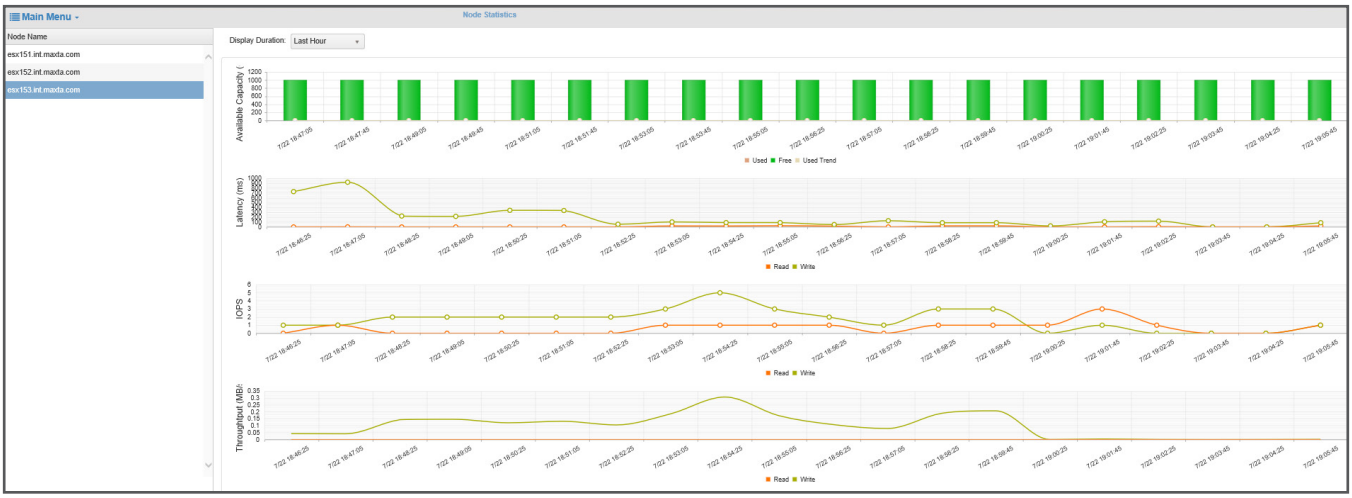


Figure 37: Node Statistics

Virtual Machine and Virtual Disk Statistics

Statistics for virtual machine and virtual disk read and write latency, as well as cache hits and misses are maintained by MxSP's statistics database and displayed within MxInsight. Data is kept for every VM which is deployed to Maxta storage, enabling users select each VM and its associated virtual disks when monitoring performance.

MxSP keeps track of latency and cache hits/misses on a per-VM and per-vDisk basis. The cache hit statistics are broken down by each component in the MxSP I/O path (as described in the I/O Path section). Information about hits in MxIL, SSD read cache, and RAM read cache, as well as misses that went to HDD, allow for a better understanding of each workload running in the Maxta cluster. As an example, a VM which has high miss percentages can attain performance improvements by using Maxta's application defined policies to change the caching policy of other VMs in the cluster.

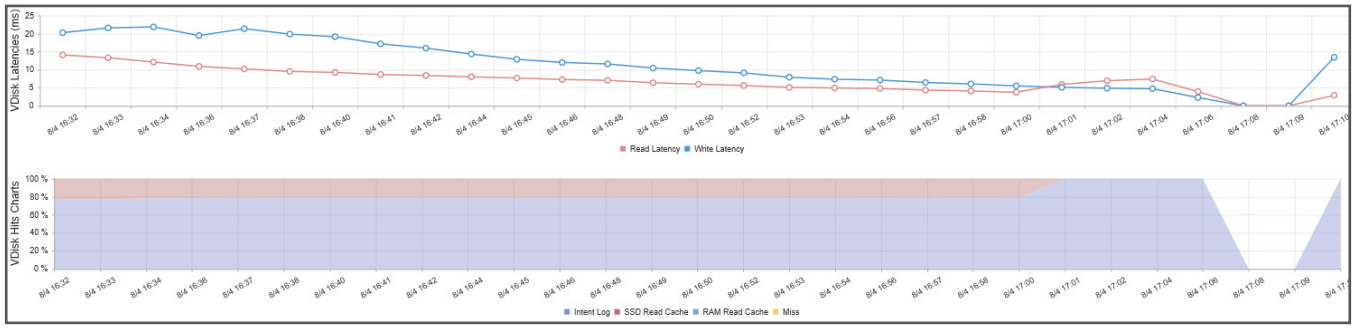


Figure 38: Virtual Disk Statistics

Storage Pool Statistics

MxInsight displays statistics for each storage pool as it relates to each node in the Maxta cluster. Storage pools within MxSP are broken down by each HDD attached to a node and the aggregate SSD resource pool for a node. The graphs shown reflect information for I/O latency and data capacity of each storage pool.

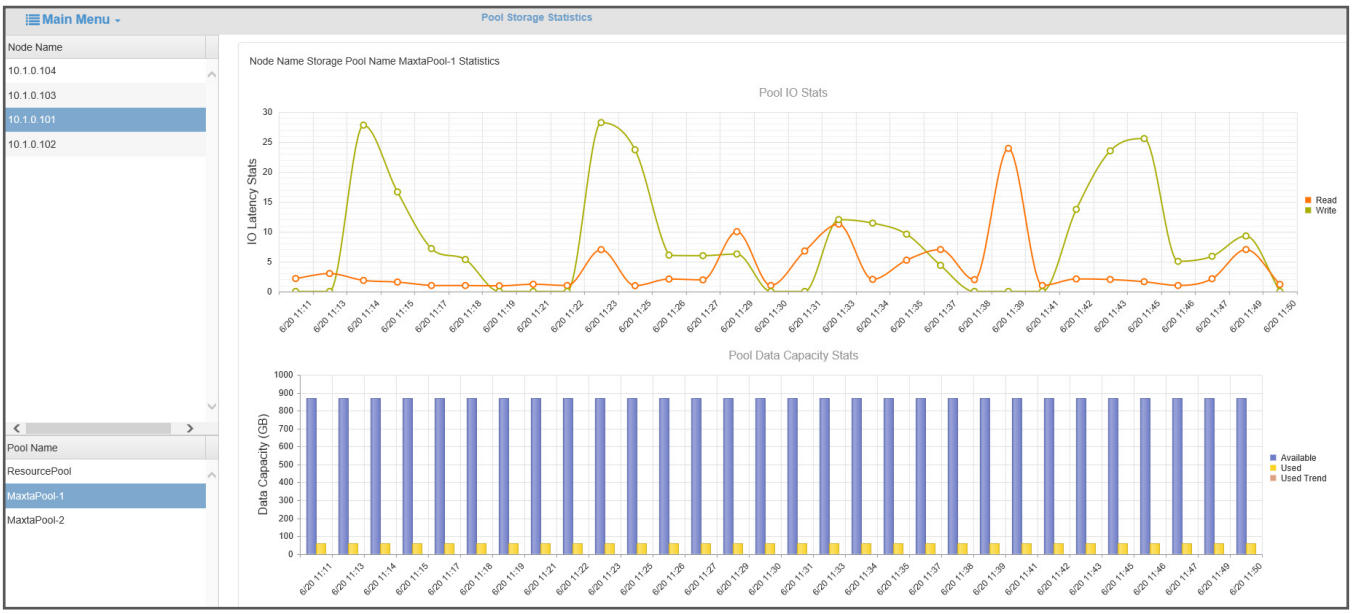


Figure 39: Storage Pool Statistics

Historical Data

Historical data can also be viewed for all statistics captured within MxInsight. The historical graphs display data collected from the last hour by default, but also have an option to select the data collection duration – ranging from one hour to one week back, with the ability to view data beyond the past week by customizing the date/time interval. Historical data can be used to assess long-term performance and determine if there are any performance trends, allowing for adjustment of workloads to maximize the cluster's performance.

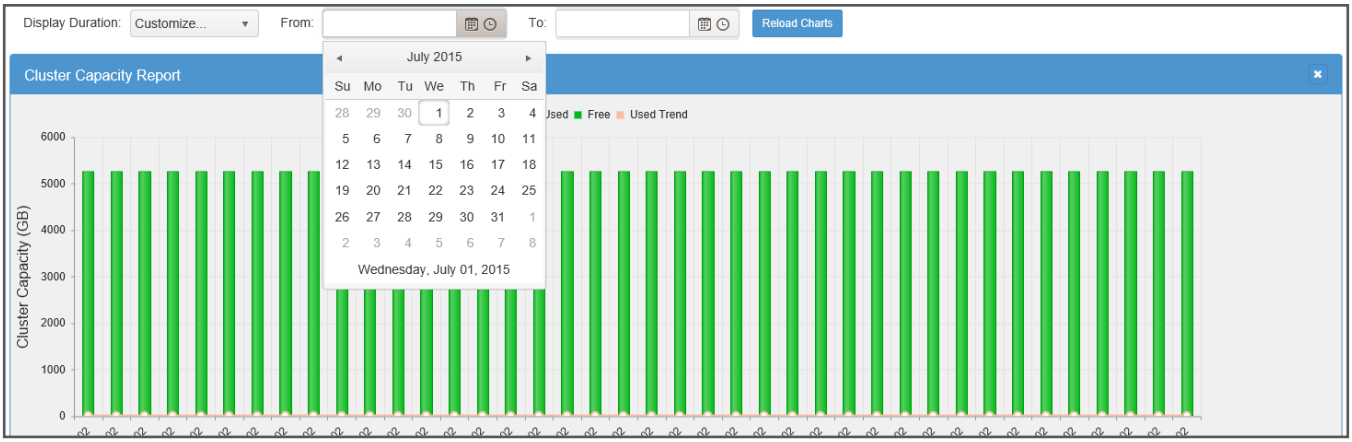


Figure 40: Historical Data for Cluster Capacity

Alerts and Call-Home

Messages in MxSP are broken down into three severities: Info, Warning, and Critical. Critical messages are classified as Alerts, and all other classifications are considered to be Events. Alerts and events are maintained in distinct log files within the Maxta infrastructure. Events include user actions on the Maxta cluster such as creating a VM, configuring snapshot policy, or adding a disk. Alerts cover the scope of node and disk state (including SSD wear), as well as capacity usage for spinning disk and flash media, among other notifications. Administrators can set the threshold for alerts to determine when these statistics have reached a critical state (e.g., 90% capacity usage).

There is an important distinction between events and alerts.

- Events are logged as changes occur on your cluster that may be of interest to you or require your attention (depending on severity). For example, an event is logged if a disk is pulled resulting in the migration of a VM replica; events are logged for the start and completion of the migration process.
- Alerts are critical events that require your action. For example, a node or disk failure will trigger an alert.

Category	State	Description
Node	ONLINE	Indicates that a node is online. This state typically occurs if a node has been marked as missing or a failed node has been replaced and brought back online.
	FAILED	Indicates that a node has failed and needs to be replaced.
	MISSING	Indicates that a node has been marked as missing from the cluster. If the node remains in the MISSING state for longer than 12 hours, it will be marked as FAILED.
	CAN BE REMOVED	Indicates that a node that was marked for removal can be safely removed.
Disk	ONLINE	Indicates that a disk is online. This state typically occurs if a disk has been replaced and brought back online.
	FAILED	Indicates that a disk has failed and needs to be replaced.
	CAN BE REMOVED	Indicates that a disk that was marked for removal can be safely removed.
Element State	ELEMENT ONLINE	Indicates that a disk is online. This state typically occurs after a degraded or faulted disk has been brought back online.

Category	State	Description
	ELEMENT OFFLINE	Indicates that a disk is offline.
	ELEMENT DEGRADED	Indicates that a disk is unhealthy.
	ELEMENT FAULTED	Indicates that a fault was detected in a disk.
Storage Pool State	POOL ONLINE	Indicates that the storage pool is online. This state typically occurs after an offline pool has been brought back online.
	POOL OFFLINE	Indicates that the storage pool is offline. The storage pool can go offline due to reboot or due to hardware failure.
	POOL DEGRADED	Indicates that the storage pool is unhealthy and may require hardware replacement.
Capacity	<%>	Indicates the percentage of used capacity.
SSD	<1-100>	Indicates the SSD wear. 100 indicates that the SSD is new. 1 indicates that the SSD is nearing end of life and should be replaced.

Figure 41: Alert List Column Descriptions

Severity	Description
Info	Provides information about a cluster resource.
Warning	Indicates that a cluster resource requires your attention. A warning severity can change to critical and generate an alert.

Figure 42: Event Severity Descriptions

Email notifications for alerts can also be configured through MxInsight. This is accomplished by configuring the appropriate SMTP settings and subsequently adding the email addresses of those who should receive notifications.

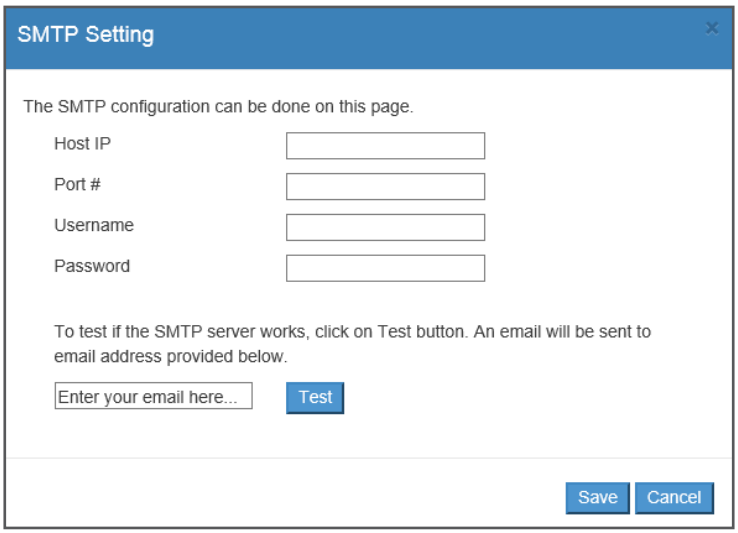


Figure 43: Configure SMTP Settings for Alert Notifications

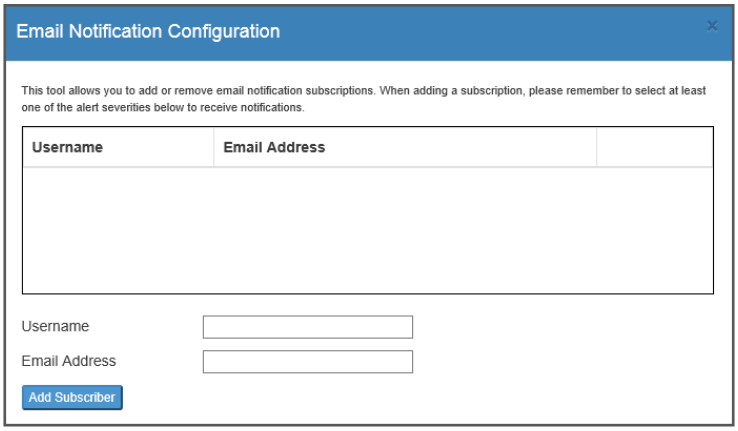


Figure 44: Add Email Users for Alert Notifications

In addition to Alerts and Events, the Maxta UI also displays information about Recent Tasks and Virtual Machine Resyncs. Recent Tasks displays all task activity (creation of VMs, snapshot creations and deletions, etc.) of the Maxta cluster. Failed tasks are retained for 2 hours while successful tasks are retained for 15 minutes. VM resyncs occurs when virtual machine replicas (mirrored copies) become out of sync due to nodes or disks shutting down or failing. As data writes continue to replicas on online nodes, the data on the offline node becomes stale. When the node is brought back online, MxSP automatically resynchronizes the data. MxInsight shows the status of each virtual machine resynchronization task on the Maxta cluster so that users are aware of their cluster's health.

MxCloudConnect

Additionally, Maxta has developed MxCloudConnect, a cloud based proactive reporting and alerting system which is included as part of the standard MxSP installation. This service provides administrators with access to a web-based portal which has information on cluster configuration, alerts, and events across multiple Maxta clusters. Maxta's support team uses MxCloudConnect to receive automated updates on the state of all MxSP deployments and respond immediately to any alerts. This quickens response time on support calls and reduces customer headaches. MxCloudConnect can also be used by managed service providers to monitor client deployments of MxSP and ensure that all clusters are functioning properly. An application key and credentials are provided by Maxta when MxSP is downloaded. This information must be entered into MxInsight in order to enable the MxCloudConnect service.

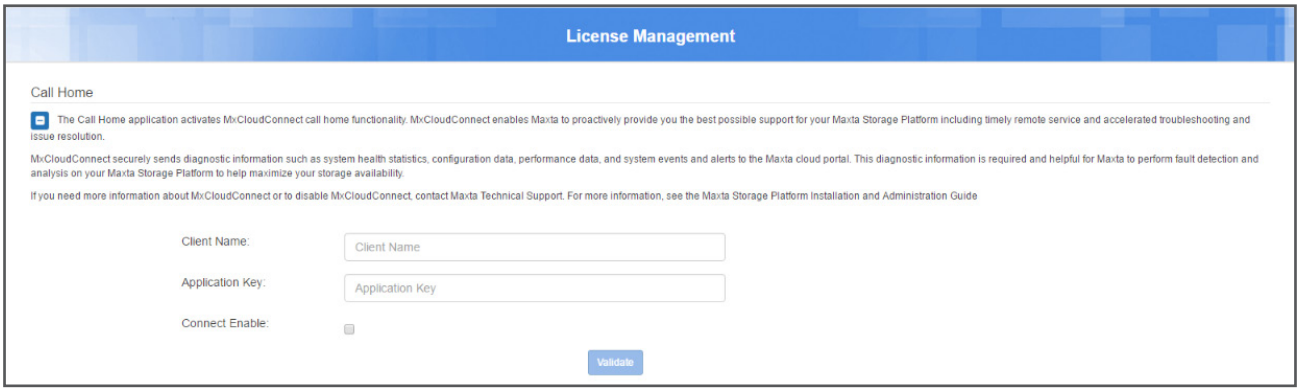


Figure 45: Enabling MxCloudConnect

MxCloudConnect leverages Maxta REST APIs to fetch configuration data, alerts, and events from the Maxta Management Server and push them to a web-based portal. Data is only sent outbound from the Maxta cluster to the Maxta cloud, and all communication occurs on a secure connection via HTTPS. Events are collected as a delta of the previous API call and are appended to the existing list in order to provide a full history of events in the Maxta cluster. Meanwhile, all alerts are collected on every API call and these alerts overwrite the existing set of data, thereby providing administrators with the full set of unacknowledged alerts at any point in time.

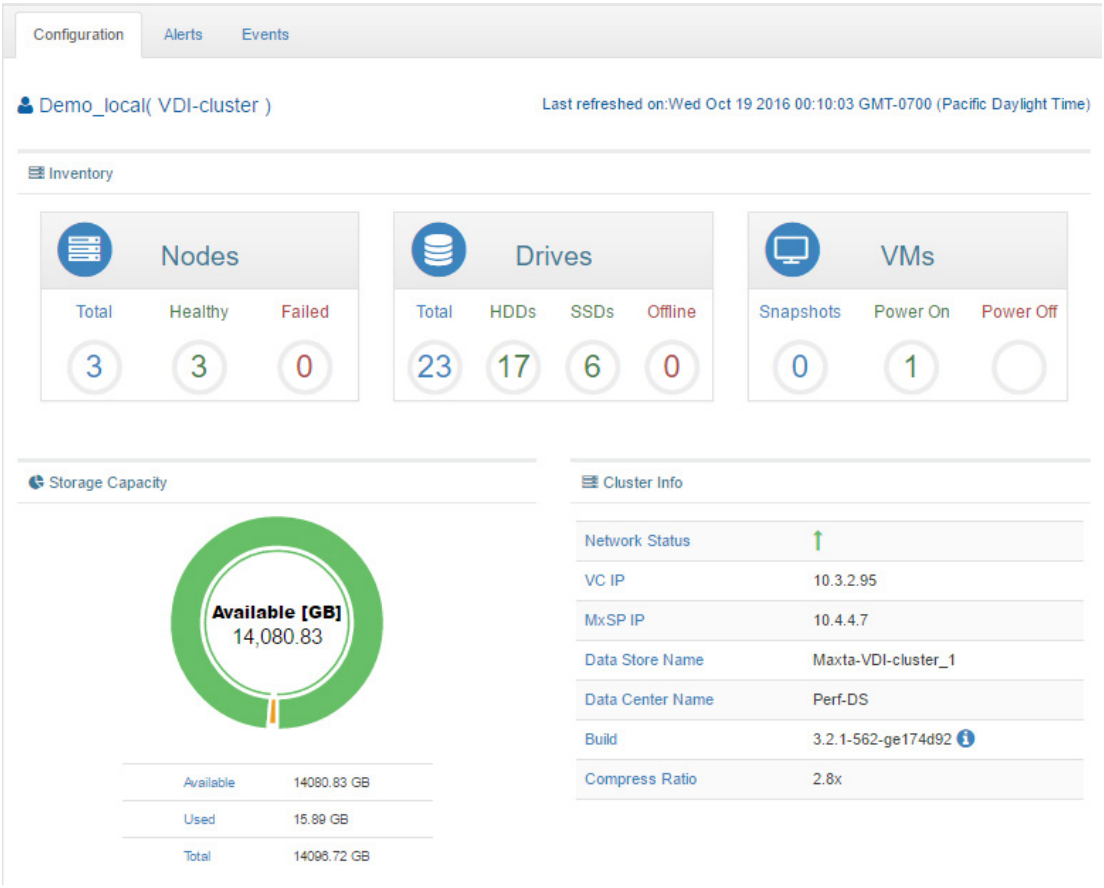


Figure 46: Configuration Data in MxCloudConnect

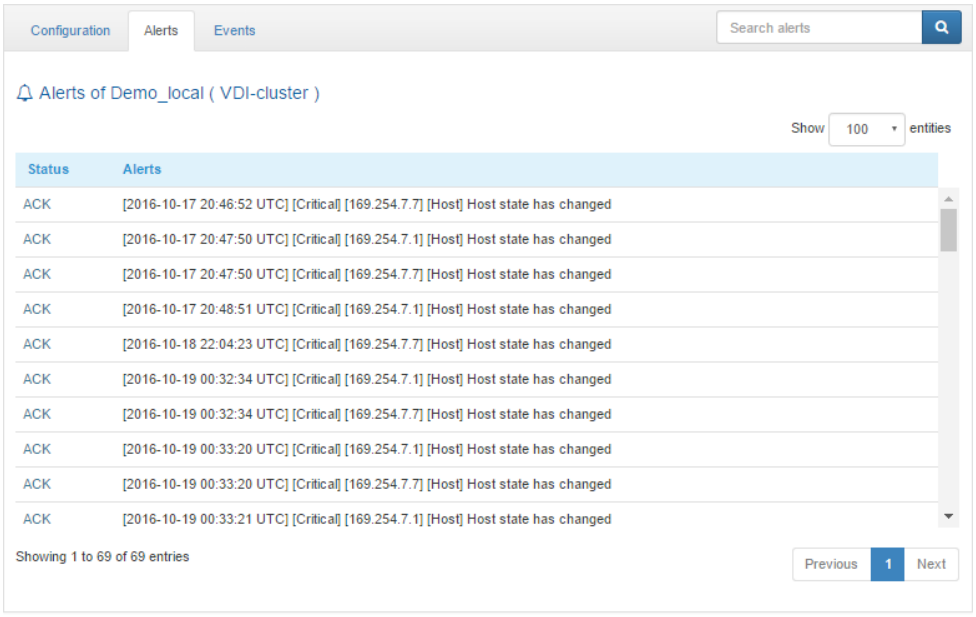


Figure 47: Alerts View in MxCloudConnect

MxCloudConnect inherits the high availability of the Maxta Management Server – when the node running MxCloudConnect (always the same node as the Maxta Management Server) fails, the Maxta Management Server fails over to another node on the cluster and starts the MxCloudConnect service on that node. Email and SMS notifications are used in the event that the Maxta Management Server fails or the MxCloudConnect service fails and administrators are unable to access the web portal.

Installation and Upgrade

Installation of Maxta software is a simple process which only requires hypervisor manager credentials in order to gather all necessary information about the cluster. Users can select which servers, drives, and networking they want to incorporate into the Maxta cluster during installation. Maxta software can also be upgraded and patched non-disruptively at no cost to cluster uptime.

Installing Maxta Storage Platform

In VMware environments, the Maxta installer is an .exe file which can be run from any Windows machine that has connectivity to the vCenter and ESXi hosts. For KVM installations, MxSP is installed via a Python script and associated configuration file. Installation of Maxta software will deploy MxSP services on each node in the cluster, configure storage networking, and create the Maxta storage. After installation has completed, users can immediately begin to deploy VMs on Maxta storage and access MxInsight to monitor storage infrastructure.

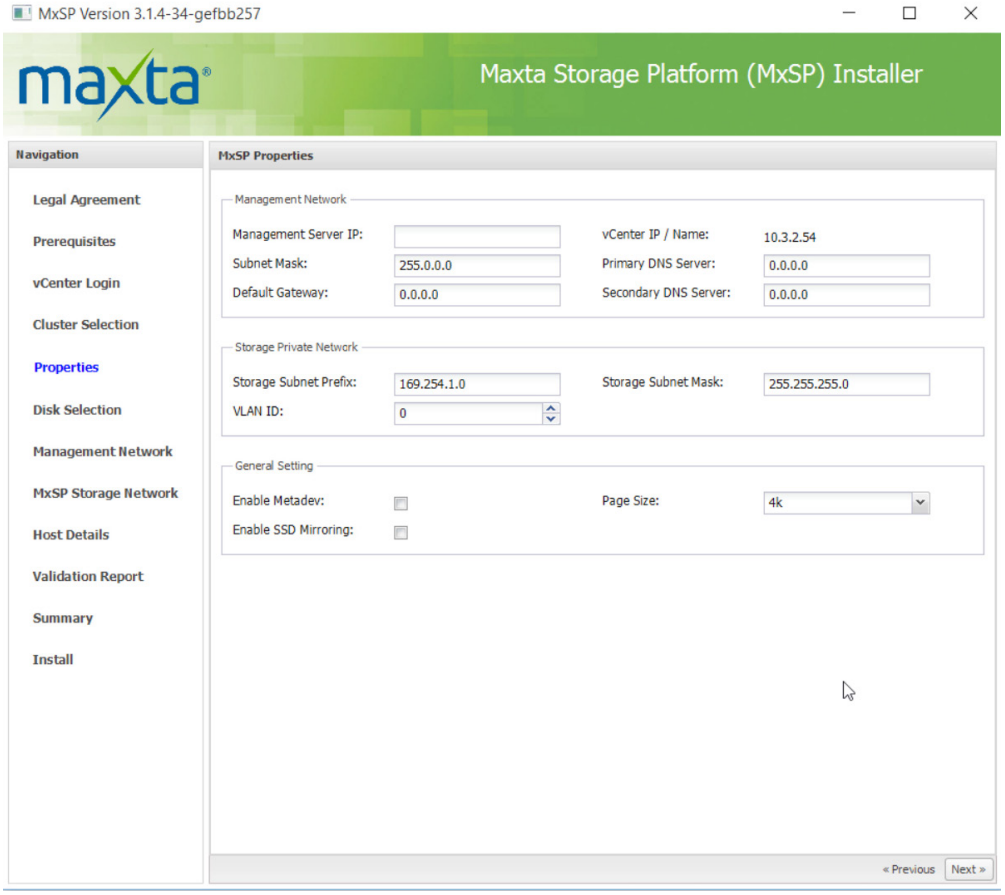


Figure 48: Configure Maxta Networking and Features

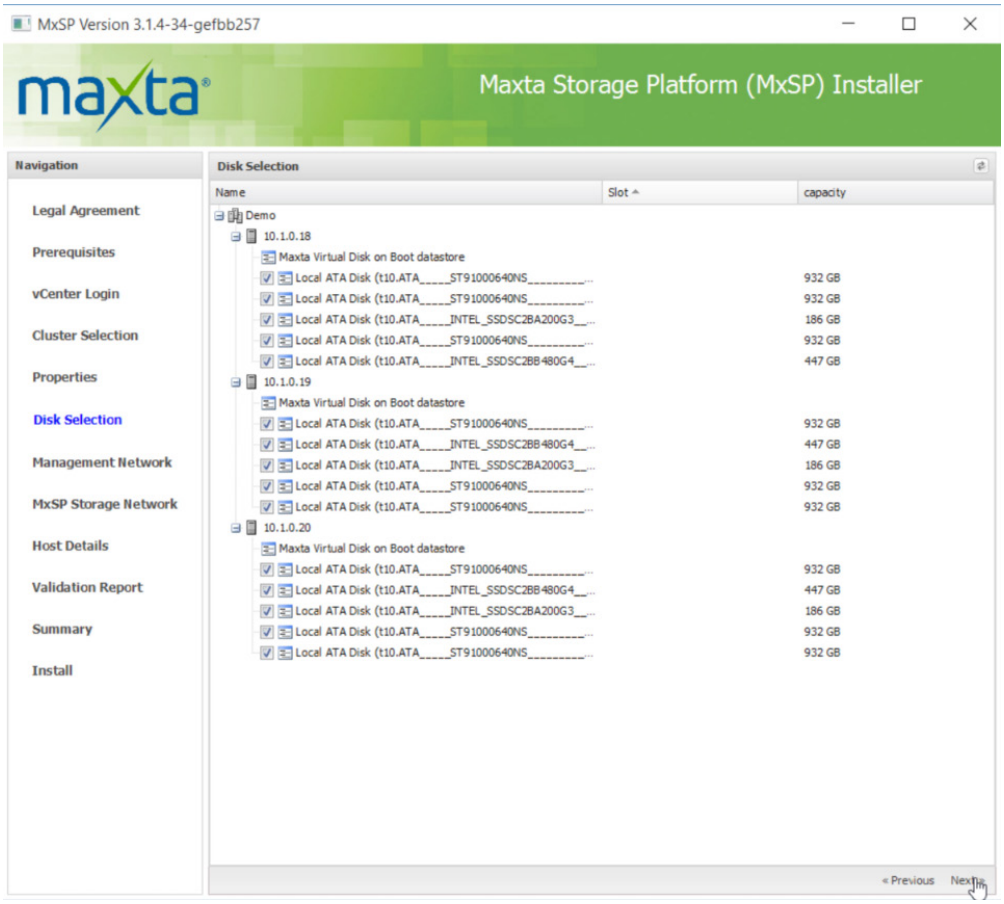


Figure 49: MxSP Installer Disk Selection

Upgrade and Patching

MxSP performs a fully automated rolling upgrade wherein the new bits are applied to each Maxta controller VM in the cluster one node at a time. When the VM goes down for upgrade, MxSP's NFS IP failover will ensure that the data is still available through another node in the Maxta cluster. Therefore, there is no need to move virtual machines off the servers in the cluster – patch and upgrade of MxSP is completely non-disruptive and requires zero downtime.

When the Maxta controller VM has been patched and the services are restarted, a replica resync will occur for all data on the affected node. MxSP's upgrade process ensures that the resyncs complete before starting the patch on the next node in the Maxta cluster. This maintains data availability by ensuring that at least one up to date copy of data is available for all VMs across the Maxta cluster.

The process for upgrading the hypervisor is very similar to that of upgrading MxSP. Replica copies will ensure that there is zero data loss, and NFS IP failover will ensure that there is no downtime in the Maxta cluster. However, with hypervisor upgrades, the administrator must wait for the replica resync to complete before upgrading another node. This is similar to the automated process that occurs when upgrading MxSP, but since the hypervisor upgrade is done outside of MxSP's visibility, the administrator must ensure that resyncs have completed before bringing down a node.

Please note that in order to maintain full support of high availability during upgrade, Maxta recommends that a policy having more than two replica copies be set for all affected VMs.

Benefits

Use Cases

The Maxta solution can be provided directly as a solution to enterprises or Managed Service Providers (MSPs) looking to increase their margins. MxSP integrates into existing frameworks for user access control and multi-tenancy, thereby allowing MSPs to enforce resource constraints as desired throughout their environment.

Business Critical Application

The flexibility of the Maxta Storage Platform enables it to support a variety of use cases in the data center. MxSP can be the storage solution for business critical applications. MxSP's full suite of enterprise class services meet the criteria necessary to support workloads running on primary storage.

Remote Office and Branch Office

MxSP is also a great fit for Remote Offices and Branch Offices (ROBO). ROBO environments require the same enterprise-class data services of primary storage, but operate within a more limited budget and face more infrastructure constraints. Maxta's focus on VM-centric management overcomes the lack of specialized skills commonly found in ROBOs. Additionally, bringing compute and storage together onto hyper-converged nodes greatly reduces space and power requirements.

Test and Development

Maxta's zero-copy snapshots and clones are advantageous to test and development teams who need point-in-time copies of the latest production data. Rather than copying an entire LUN, as is the model for traditional storage arrays, Maxta's snapshots and clones are taken at the VM-level. The snapshots and clones are also time-, performance-, and capacity-efficient, meaning that they can be created and deleted instantly, have no impact on performance, and do not take up any space on creation.

Virtual Desktop Infrastructure

Maxta's snapshots and clones are also highly beneficial to Virtual Desktop Infrastructure (VDI) deployments in which administrators have to deploy thousands of virtual desktops in minutes. These environments have to maintain a better end-user experience than a traditional stand-alone desktop at lower cost. MxSP is able to meet the performance requirements of VDI by leveraging SSDs as read and write cache while providing a cost-efficient solution.

Disaster Recovery

Maxta storage can serve as an endpoint in Disaster Recovery (DR) environments. MxSP leverages the existing replication software (Zerto, Veeam, etc.) to provide VM-level recoverability. The recovery site maintains all of the enterprise-class capabilities of MxSP, including the ability to maintain short term backups via Maxta's snapshots.

Increase Flexibility

Maxta delivers on the promise of hyper-convergence by providing flexibility and choice at all levels of the datacenter. MxSP's hypervisor agnostic architecture allows administrators to pick the hypervisor of their choice. The software-defined approach provides flexibility when it comes to servers and disks, enabling customers to deploy Maxta's hyper-converged solution on the hardware of their choice. Maxta leverages flash media installed on commodity servers and disks to maximize performance while minimizing cost. MxSP eliminates the vendor lock-in commonly associated with appliance based hyper-converged solutions. Additionally, Maxta provides flexibility in purchasing the software through a one-time upfront perpetual license or a recurring subscription-based service.

Simplify IT Management

MxSP dramatically simplifies storage management by eliminating the need for provisioning storage or managing volumes, LUNs, file systems, RAID, ACLs, etc. This removes the necessity to develop the mapping between virtual machines and storage constructs such as LUNs and volumes, minimizing administrative errors. Converging compute and storage on standard servers eliminates storage networking tasks such as zoning. MxSP provides a single-pane-of-glass for both VM and storage administration by incorporating all storage functionality within the virtualization UI. Once MxSP is installed, the administrator can simply point to the Maxta storage pool from the virtualization UI during VM creation and MxSP will take all steps to optimally provision and allocate resources for the new VM.

The installation and configuration of MxSP takes only a few minutes, enabling all system/VM administrators to install and manage storage. During installation, MxSP aggregates storage across all the servers in the cluster and presents it as the Maxta storage pool with a global namespace that is used to provision storage for VMs. MxSP will not add any previously allocated storage devices to the Maxta storage pool. It is also possible to selectively exclude any storage device from

the Maxta storage pool. The installer automatically installs and configures all the necessary pre-requisite software and identifies the minimum hardware and software requirements for installing MxSP. The installer guides the user through the resolution process in case the minimum requirements are not met.

The Maxta storage pool and all data services such as replication, snapshots, and zero-copy clones are configured and managed from the VMware vSphere Client at VM granularity. This simplification eliminates the day-to-day tasks of storage management and enables administrators to focus on managing applications and VMs.

Application Defined

Traditional storage arrays offer the ability to define storage on a per-application basis through LUN configurations. However, this is a very complex process which requires investment in heavily specialized storage administrators. Hyper-convergence has relieved this pain by simplifying management and deployment. Unfortunately, by creating a single, uniform pool, other hyper-converged solutions have lost the flexibility to configure storage for each application – one size is NOT good for all.

Maxta brings the best of both worlds with its application defined storage platform providing both ease of management and the ability to customize storage to individual applications. Through Maxta's management interface, users can easily set policies with just a few clicks on a per-VM or per-vDisk basis. These policies help to configure each application for failure tolerance and data layout, among other characteristics.

Linear Scalability

The versatile architecture of MxSP provides the ability to scale capacity and performance independently on-demand without having to over-provision resources. MxSP also delivers linear scalability as new servers are added to the cluster without impacting performance. The example below highlights the linear scalability of the platform in a VDI use case. The average latency of accessing the virtual desktops remained almost the same even after increasing the number of VMs deployed and adding nodes to the cluster. The linear scalability of MxSP from 100 to 456 desktops without impacting performance is shown below:

Maxta Storage Platform (MxSP)			
Number of Desktops	100	300	456
Number of Nodes (ESXi hosts)	2+1*	3	4
Number of stuck or unresponsive desktops (Sessions)	0	0	0
Controller VM configuration (CVM)	4vCPU/8GB Memory	4vCPU/8GB Memory	4vCPU/8GB Memory
Performance Metrics			
Minimum response time (VSIbase)	731 ms	797ms	843 ms
Average response time (VSImax Average)	823 ms	929 ms	995 ms
Maximum response time (VSIMax Threshold)	1731 ms	1798 ms	1844 ms

*** 2 Hyper-Converged Compute/Storage Nodes + 1 Compute-Only node**

Figure 50: MxSP Scales Linearly in VDI Deployments

Maximize Savings

MxSP delivers the capabilities of storage arrays in software by leveraging server-side SSD and internal drives or JBODs connected to standard servers. MxSP enables significant CAPEX savings by converging compute and storage resources on standard servers leveraging commodity components, without compromising performance or scalability.

MxSP eliminates all storage provisioning activities and dramatically simplifies day-to-day data management activities such as recovery of virtual machines from snapshots and creation of new VMs rapidly using clones. Additionally, MxSP eliminates storage management tasks such as changing RAID parameters or storage tier management. MxSP provides significant reduction in power, cooling, and floor space by converging compute and storage resources on standard commodity servers and eliminating the need for storage arrays.

Storage Efficiency

MxSP delivers capacity optimization capabilities such as thin provisioning, in-line compression, and in-line de-duplication to enable significant storage efficiency. Thin provisioning provides the ability to provision capacity several times larger than the amount of physical storage resources. This enables procuring incremental capacity on demand. MxSP allocates space on a per-page basis – if the page size for a VM is set to 4K, MxSP will only allocate 4K chunks of data. This minimizes any possible overprovisioning and maximizes the amount of capacity savings. Thin provisioning is enabled by default, and does not require an upfront reservation of capacity.

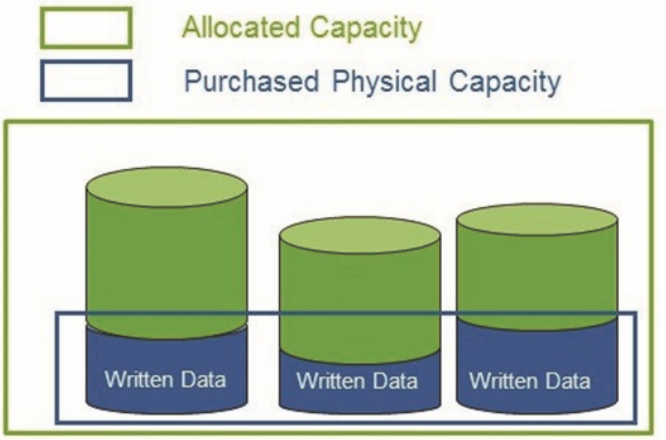


Figure 51: MxSP Maximizes Capacity Savings

Compression is also enabled by default. MxSP performs in-line compression of data and metadata with minimal performance penalty. Data and metadata are compressed while de-staging from flash to spinning disk, but are not compressed in MxIL and SSD read cache in order to improve latency. MxSP's default policy utilizes the LZ4 compression algorithm to provide improved storage efficiency without impacting performance and CPU utilization. The combination of compression and thin provisioning provide the average Maxta deployment with 2-4X in capacity savings.

MxSP also provides users the ability to perform in-line de-duplication. This feature is not enabled by default on the storage platform. Maxta recommends leaving de-duplication disabled since it is resource intensive and can cause performance degradation.

Software Agility

The traditional storage model packages hardware together with non-transferable software. When storage hardware is refreshed every four to five years, software must be purchased again. Hyper-converged appliances are no different in this model of non-transferable software. In fact, this approach forces end-users to make a compromise due to the faster pace of innovation in compute technology compared to that of storage. Adopters of hyper-converged appliances must either: refresh compute at a slower pace (resulting in hardware which is out of date and slower renewal rates for partners) or refresh storage at a faster pace (creating a higher storage acquisition cost for the customer).

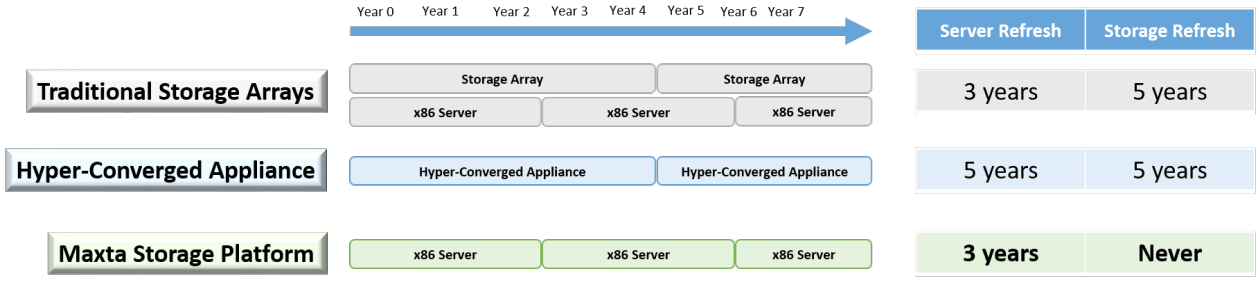


Figure 52: Software Agility

Maxta de-couples the software from the hardware with its fully transferable software license. This enables end-users to purchase the hardware they need, when they need it, without having to repurchase storage software – Maxta is the last storage software purchase in the datacenter.

Conclusion

Maxta is a hypervisor agnostic implementation of software defined storage supporting VMware vSphere, KVM, and OpenStack. The increased flexibility enables customers to select their server of choice. Maxta simplifies administration and minimizes the learning curve by integrating into the virtualization management interface. MxSP dramatically simplifies IT by eliminating the need for provisioning storage and managing volumes, LUNs, file systems, and RAID. The installation and configuration of MxSP takes only a few minutes. Additionally, all data services, such as replication, snapshots, and zero copy clones, are configured and managed at the VM level rather than the storage level. This enables the VM administrator to leverage storage without the need for deep storage and vendor specific expertise. With MxSP, administrators can manage VMs and not storage.

MxSP delivers enterprise-class data services with best-in-class resiliency, continuous availability, data protection, and agility. With unlimited VM level snapshots, backup performance is improved and recovery times are dramatically reduced. By leveraging VM level clones, rapid provisioning of VMs is possible. With continuous data protection, disaster recovery of applications is greatly simplified and accelerated for business continuity. MxSP seamlessly integrates with all the advanced capabilities of the server virtualization software.

Maxta's MaxDeploy appliances deliver a predefined and pre-validated solution that removes interoperability and performance guesswork and simplifies the ordering process.

MxSP enables significant capital savings by converging compute and storage resources on standard commodity servers, without compromising performance or scalability. This provides considerable up-front capital savings compared to storage arrays. In addition, MxSP leverages any combination of magnetic disk drives and flash technology, snapshots, zero-copy clones, thin provisioning, in-line compression, and in-line de-duplication to increase storage efficiency and reduce storage expenses. By significantly simplifying IT, increasing IT efficiency, and enabling administrators to focus on managing applications and VMs, MxSP enables dramatic reduction in operating expenses.