SCALE COMPUTING

SC//HyperCore and SCRIBE

Theory of Operations

HyperCore Version 9.1+

Technical White Paper

About this Document

Scale Computing Platform combines everything you need: virtualization, servers, storage, and backup/disaster recovery under powerful fleet management to deliver a single manageable solution for edge computing.

Intentions

This document is intended to describe the technology, concepts and operating theory behind the Scale Computing HyperCore™ OS that powers it, including the SCRIBE (Scale Computing Reliable Independent Block Engine) storage layer.

Technical Support and Resources

There are many technical support resources available for use. Access this document, and many others, at www.scalecomputing.com/support.

Online Support

You can submit support cases and view account information online through the Portal at www.scalecomputing.com/support. You can also Live Chat with support through www.scalecomputing.com during standard hours Mon-Fri 8 AM to 6 PM ET.

Telephone Support Support

Telephone Support Support is available for critical issues 24/7 by phone at 1-877-722-5359 in the US and at 0808 234 0699 in Europe. Telephone support is recommended for the fastest response on priority issues.

Professional Resources

Scale Computing offers many professional service options for both remote and on-site assistance in getting your cluster up and running quickly and knowledgeably. Contact your Scale Computing sales representative today to discuss our service offerings. Find additional information at www.scalecomputing.com/professional-services-and-programs.

Document Revision History:

Version 3.1: 05/2022	Content review and design overhaul for product rebrand
Version 3.1: 05/2021	Minor edits for Versions 8.5+
Version 3.0: 01/2020	Content review and design overhaul for corporate rebrand

Table of Contents

Design Goals

SC//HyperCore and the Scale Computing Reliable Independent Block Engine (SCRIBE) storage layer were designed to provide highly available, scalable compute and storage services while maintaining operational simplicity through highly intelligent software automation and architecture simplification.

SC//HyperCore adds intelligent automation to the hypervisor and storage layers and was designed to take advantage of low cost, easily replaceable, and upgradeable “commodity” hardware components including the virtualization capabilities built into modern CPU architectures. By clustering these components together into a single, unified, and redundant system, the architecture creates a flexible and complete “data center-in-a-box.” SC//HyperCore operates as a redundant and elastic private “cloud” that scales seamlessly with the automatic incorporation of additional nodes and handles hardware failures gracefully with minimal effort or disruption.

SC//HyperCore and SCRIBE Overview

SC//HyperCore is based on a 64-bit, hardened and proven OS kernel and leverage a mixture of patented proprietary and adapted open source components for a truly hyperconverged product. All components—storage, virtualization, software and hardware—interface directly through the SC//HyperCore hypervisor and SCRIBE storage layers to create an ideal computing platform that can be deployed anywhere from the data center to the edge of the network.

Software Defined Storage

A critical software component of SC//HyperCore is SCRIBE. SCRIBE is an enterprise-class, clustered, block storage layer that is purpose-built to be consumed by the KVM-based SC//HyperCore hypervisor directly. SCRIBE discovers all block storage devices—including flash-based solid state disks (SSDs) and conventional spinning disks (SATA or SAS)—and aggregates the block storage devices across all nodes of SC//HyperCore into a single managed pool of storage. All data written to this pool is immediately available for read or write access by any and every node in the storage cluster, allowing for sophisticated data redundancy, load balancing intelligence, and I/O tiered prioritization.

More unique aspects of SCRIBE, including how it makes efficient use of flash storage where available for tiered data placement based on historical heat mapping, are detailed in the SC//HyperCore SCRIBE—Mirroring, Tiering and Wide Striping section.

Software Managed Compute

The SC//HyperCore software layer is a lightweight, type 1 (bare metal) hypervisor that directly integrates into the OS kernel and leverages the virtualization offload capabilities provided by modern CPU architectures. Specifically, SC//HyperCore is based on components of the KVM hypervisor which has been part of the Linux mainline kernel for many years and has been extensively field-proven in large-scale environments.

SC//HyperCore integrates the SCRIBE storage pool directly into the KVM hypervisor. This means that virtual machines (VMs) running on SC//HyperCore have direct block-level access to the SCRIBE “virtual storage device” (VSD) virtual disks in the clustered storage pool without the complexity or performance overhead introduced by using remote storage protocols and accessing remote storage over a network (although the backplane network is used to communicate data locally between the cluster nodes).

Unlike other seemingly “converged” architectures in the market, the SCRIBE storage layer does not run inside a virtual machine as a VSA or controller VM but instead runs parallel to the hypervisor, allowing direct data flows to benefit from zero-copy shared memory performance—also unlike other architectures.

SCRIBE is not a re-purposed file system with the overhead introduced by local file / file system abstractions such as virtual hard disk files that attempt to act like a block storage device. This means that performance killing issues such as disk partition alignment* with external RAID arrays become obsolete. Arcane concepts like storing VM snapshots as delta files that later have to be merged through I/O killing brute-force reads and re-writes are also a thing of the past with the SCRIBE design.

SC//HyperCore—The Hypervisor Simplified

SC//HyperCore bundles a variety of adapted open source and proprietary, intelligent software to create a simplified operating system. Custom built utilizing the KVM architecture to integrate with the SCRIBE storage layer directly, SC//HyperCore makes virtualization and software automation look easy—and it is.

Create a Virtual Machine

Creating a virtual machine is a simple process that is accomplished through the SC//HyperCore interface in a single dialog box. The interface is also used to upload Operating System and Virtual Appliance ISO installation images for VM installation as part of the VM creation process (ISOs are virtual DVD / CD images). ISO images can be uploaded by drag-and-drop anywhere on the SC//HyperCore UI, or you can manage and upload ISOs through the media tab in the SC//HyperCore Control Center console. Once uploaded, ISO images are available for use by any future VMs.

Selecting the Create VM option (the + icon in the UI) right from the main VM management screen allows the user to specify required and optional parameters for the virtual machine including:

- VM name and optional description
- Optional tags that allow logical VM grouping (the VM name, description and tags are searchable and filterable in the main VM management screen)
- Number of virtual CPU cores
- VM RAM
- Number and size of virtual, thin provisioned disks
- A previously uploaded virtual DVD/CD ISO image for installing an operating system

Creating a virtual machine not only creates persistent VM configuration parameters but also creates virtual disks using the SCRIBE distributed storage pool that attach to the VM when started. SC//HyperCore VMs are able to access their virtual disks directly as if they are local disks, without the use of any SAN or NAS protocols, regardless of which node the VM is running on at the time.

SC//HyperCore virtual disks are thin provisioned so that virtual disk space is not fully allocated until it is actually used by the virtual machine. On systems that include tiered storage resources such as SSD and spinning disks, SC//HyperCore assigns each virtual disk a default "flash priority" setting designed to provide the ideal balance of performance versus storage cost for every VM. Flash priority and tiered clusters are discussed in further detail in the SC//HyperCore SCRIBE—Mirroring, Tiering and Wide Striping section.

SC//HyperCore automatically creates a single virtual NIC for each VM but the administrator can easily add additional virtual NICs and/or specify VLAN tags to control the VMs access to network resources.

* Virtual disk block alignment problems can happen when you try to represent a virtual block device as a file sitting on a file system with its own block size which itself will sit on top of some other physical or logical block storage that may have its own block size and physical data layout.

Virtual Machine Placement

Because all SC//HyperCore capable nodes have access to the entire pool of storage, virtual machine placement on nodes is determined by the availability of compute resources (RAM and CPU).

For example, if a new VM requires 16 GB RAM, SC//HyperCore selects a node with at least 16 GB RAM available automatically. SC//HyperCore will try to determine the best available node based on the most available capacity for new VM placement. SC//HyperCore allows running VMs to be live migrated to other SC//HyperCore nodes without the VM being shutdown and with virtually no noticeable impact.

Virtual Machine Failover

If an SC//HyperCore node was running VMs prior to a node failure, those VM disks and data are still available to the remaining cluster nodes since the cluster uses a single pool of redundant storage. This allows VMs to be automatically restarted within minutes of the node failure on the remaining available nodes with SC//HyperCore automatically placing VMs on nodes with the correct availability of compute resources.

VM-Level Snapshots

SC//HyperCore offers the ability to take multiple point-in-time consistent snapshots of virtual machine storage devices (virtual disks) and the VM configuration—a complete image of the VM at that time for backup purposes. Snapshots occur nearly instantly with virtually no impact on production workloads, even with thousands of snapshots.

SC//HyperCore snapshots use a space efficient allocate-on-write methodology where no additional storage is used at the time the snapshot is taken, but as blocks are changed the original content blocks are preserved, and new content written to freshly allocated space on the cluster.

Over time, the space consumed by a particular snapshot represents only the blocks that are unique to that snapshot point-in-time – which given multiple possible snapshots would be a subset of the data that has changed since the time the snapshot was taken. Blocks that are the same in multiple point-in-time snapshots or in the current live image are reference counted so that they will be retained as long as they are referenced by the current image or any point-in-time snapshot in order to preserve cluster capacity.

This allocate-on-write methodology does not introduce the additional I/O associated with older copy-on-write snapshot methods which for every change must first read the original content, write it to a new temporary location, update the snapshot metadata to reflect that new location, then allow the new data to be written and metadata updated.

VM Snapshot Promote and Revert

SC//HyperCore VM level snapshot can be promoted to a live VM and started nearly instantaneously from the HyperCore UI by leveraging the VM cloning capability. If the goal is to revert the running VM to a previous point in time, for example to undo a problem update or patch, you can simultaneously clone the previous snapshot to a live VM and start that cloned VM while you power down and optionally delete the original problem VM. Or, retain the problem VM image and start it isolated from the production network to do a postmortem analysis.

When promoting or reverting a VM, the entire VM configuration and its snapshotted storage objects are made available, allowing you to edit the VM configuration as you choose and start it up immediately with no data recovery time required.

VM Snapshot-Based Thin Cloning

It may be obvious from the examples above, but VM Snapshots allow for instant space efficient cloning or the creation of “template” VMs. In a single step, you can clone a live, running VM by taking a snapshot and immediately promoting the snapshot to a live image. It is then possible to start the cloned VM in an isolated network and do a trial run of a new software patch, for example.

Golden “master” template VMs can be created with the latest OS, application patches, security and anti- virus tools and customized settings. These template VMs can be used to quickly deploy large numbers of similar server or desktop OS VMs. Each VM created from a snapshot clone originally takes zero additional space and grows based only on the changes made once that clone VM is booted.

VM Replication and Recovery

In addition to being highly available within a SC//HyperCore cluster, VMs can be replicated between clusters for remote availability. This integrated replication protects against additional disaster scenarios such as site and regional disasters which may impact an entire SC//HyperCore cluster, affecting local recovery options.

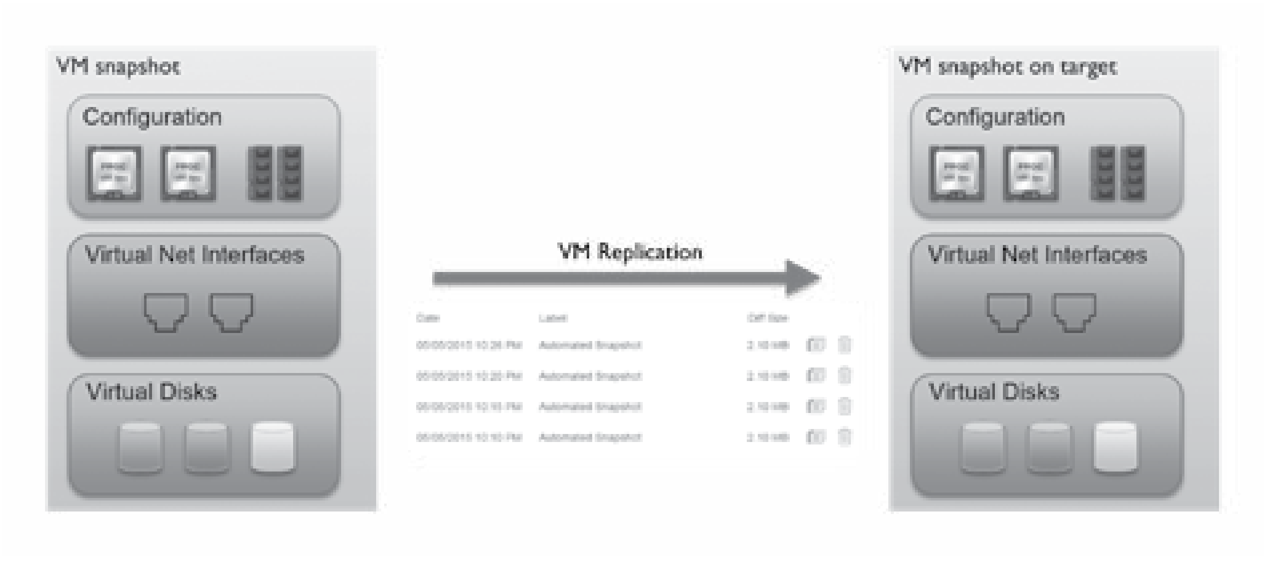
SC//HyperCore replicated VMs and the snapshots used for replication can be manually activated as live VMs with the click of a button. The recovery process from a remote replicated cluster is the same as recovering from a snapshot on the original cluster: promote a clone VM from the VM replication card image on the remote cluster.

The entire replication process can later be reversed to replicate and return one or more VMs back to the original cluster to resume operations. As with the original replication, SC//HyperCore can determine the block differences between multiple points in time which can minimize the amount of data that needs to be sent back to the original cluster.

Individual File Recovery

To recover individual files that may have been deleted or corrupted accidentally, Virtual Disk Cloning allows individual virtual disks to be cloned and mounted from snapshots so files can be recovered from a previous point in time. A virtual disk clone can be mounted to the same VM it is being cloned from or any other VM for recovery.

The virtual disk is cloned and mounted as a single action, making it easy to access the files on that disk from a live virtual machine. The virtual disk can be cloned from any available snapshot, ensuring files can be recovered as far back as snapshot are retained.



SC//HyperCore SCRIBE—Mirroring, Tiering, and Wide Striping

SCRIBE is the storage management layer embedded in SC//HyperCore and treats all storage in the cluster as a single logical pool for management and scalability purposes. The real benefits of SCRIBE come from the intelligent distribution of blocks redundantly across the cluster to maximize availability and performance for the SC//HyperCore virtual machines.

Unlike many systems where every node in a cluster are required to have local SSD—with significant dedication of SSD for metadata, caching, write buffering and/or journaling—SC//HyperCore was designed to operate with or without SSD. SC//HyperCore allows a mixing of node configurations which may or may not include SSD in addition to spinning disks. Regardless of the configuration, SC//HyperCore is able to use wide-striping to distribute I/O load and access capacity across the entire SC//HyperCore cluster, achieving levels of performance well beyond other solutions on the market with comparable storage resources and cost.

To best utilize the SSD storage tier, SCRIBE is able to assess the user configured and natural day-to- day workload priority of data blocks (discussed in more detail in the Tiering section below) based on recent data history at the VM virtual disk level. Intelligent I/O heat mapping identifies heavily used data blocks for priority on SSD storage. SC//HyperCore virtual disks can be configured individually at varying levels of SSD prioritization so that an always changing SQL transaction log disk versus a more static disk in the same VM, for example, will allocate SSD appropriately as needed for improved performance.

Data Placement

The SCRIBE storage layer allocates storage for block I/O requests in chunks ranging from as large as 1 MB to as small as a 512 byte disk sector. SCRIBE ensures that two or more copies of every chunk are written to the storage pool in a manner that not only creates the required level of redundancy (equivalent to a RAID 10 approach) but also aggregates the I/O and throughput capabilities of all the individual disks in the cluster, commonly known as wide striping.

In multi-drive SC//HyperCore clusters, all data is replicated twice on the cluster for redundancy. Data writes between the node where the VM issues the I/O request and the nodes where the blocks will be stored occurs over the private backplane network connection to isolate that traffic from incoming user/server traffic.

For example 10 GB of data is being written to a cluster of 3 nodes which contain a total of 3 solid state disks and 9 spinning disks. Each node contains 4 total disks of various available capacities. In this example each node has one 960 GB SSD and three 4 TB spinning disks, thus contributing 12.96 TB of RAW storage capacity per node or 6.48 TB of effective usable storage space to the cluster after accounting for 2 copies of all data blocks being stored. That 10 GB of data (or 10240 MB) would require 10240 chunks (1MB each) to store the first copy plus 10240 chunks to store a second copy of each chunk. Not factoring in other I/O that may be occurring at the same time, you could picture each of the 12 disks doing approximately 1700 x 1MB writes in order to store that 10 GB redundantly across the SC//HyperCore cluster.

Of course, other I/O is likely going on in the cluster at the same time, further leveraging the I/O capabilities and throughput of all the disks in the cluster. This means that the more nodes you add to the cluster, the more aggregate performance you are able to achieve. The benefits of wide striping across a large number of disks apply to both read and write I/O.

Contrast this methodology to a conventional RAID array controller where you pick a small number of disks, allocate them to a specific RAID set, and are limited to the performance of the disks in that specific RAID set. If disks in another RAID set happen to be idle, those disks could be sitting there doing nothing while disks in a heavily utilized RAID set have become a bottleneck for that application (note that there are many other common limitations of conventional RAID such as the overhead of parity calculations, rebuild times, dedicated hot spares, etc., that are beyond the scope of this document).

Review the The SC//HyperCore Cluster in Action section below for examples of how the SC//HyperCore system utilizes the SC//HyperCore and SCRIBE features in real world scenarios.

Tiering

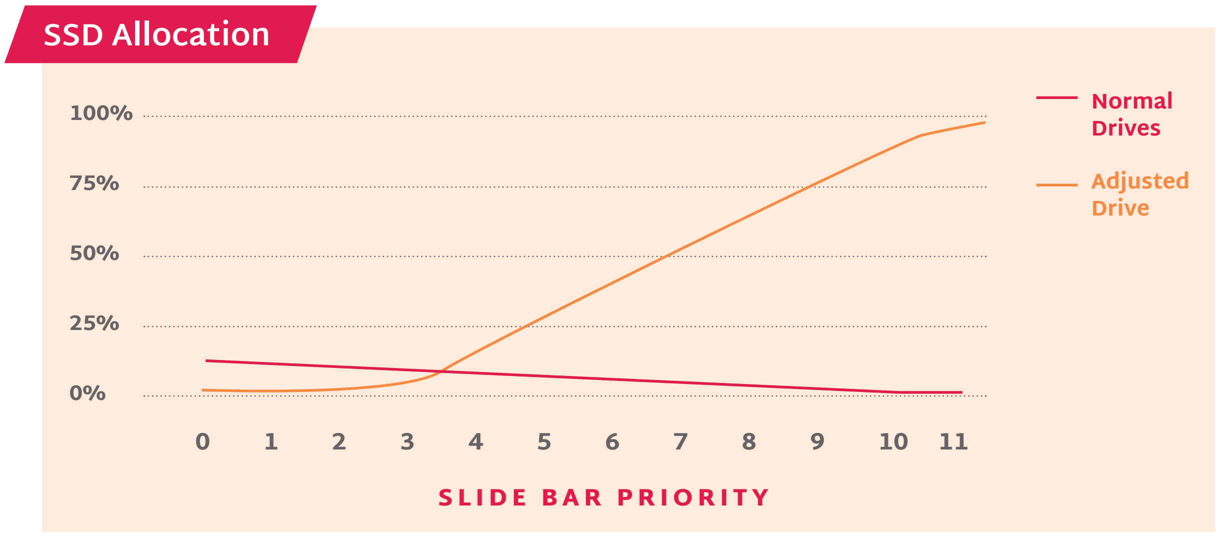
As previously mentioned, the addition of SSD storage in SC//HyperCore nodes—known as Tiered SC//HyperCore Nodes—allows more capabilities in the SCRIBE storage architecture. Tiered SC//HyperCore Nodes in conjunction with the version 7.0 and later HyperCore firmware release allows the use of the HyperCore Enhanced Automated Tiering (HEAT) feature.

HEAT includes configurable SSD priority allocation at the individual virtual disk level through an easy-to- use slide bar in the SC//HyperCore interface, and intelligent data block priority based on block I/O heat mapping assessed utilizing historical I/O information on each virtual disk.

HEAT-capable clusters will automatically assign all new writes to the SSD tier until SCRIBE is able to accurately assess their activity. In theory, an SC//HyperCore cluster with low storage utilization could be running completely on SSD.

By default, a new virtual disk will automatically get assigned a SSD priority of 4 on a scale of 0 to 11. The scale, represented by the slide bar in the HyperCore interface, is exponential, meaning that changing the priority of a VM virtual disk from 4 to 5 in the HyperCore UI doubles the priority of that data for SSD placement. This scaling means that even one position change on the slide bar can provide significant performance improvement and requires less experimenting with various settings. The values of 0 and 11 on the slide bar are unique. A setting of 0 eliminates SSD storage usage on that virtual disk (convenient for a static FTP drive, for example) and 11 changes the SSD priority by an order of magnitude, multiplying the priority of the virtual disk data for SSD placement by 10. When you turn it to 11, you really crank it.

The graph below shows an example of a VM virtual disk being prioritized from 0 to 11, assuming all other VM virtual disks remain at the default setting of 4.



Storage Pool Verification and Orphan Block Detection

For temporary failures such as a node power outage, the SC//HyperCore cluster will automatically verify that any data blocks that the node contains were not changed elsewhere on the cluster while the node was offline. If the blocks were changed, SCRIBE will clean up the original blocks in the shared storage pool if they are no longer required (possibly blocks that were held by
a snapshot).

As an example of orphan block detection, block 200 is contained on node 1 and 2. Node 1 goes offline due to a power outage and block 200 changes during the time node 1 is offline. Block 200 is then updated on node 2 and a new redundant copy written elsewhere on the cluster (as node 1 is still offline and all new or changed data always requires a redundant copy to be stored). When node 1 comes back online, SC//HyperCore will detect that the original contents of block 200 are no longer current or needed and correct that discrepancy by releasing them back into the storage pool.

The SC//HyperCore Cluster in Action

The best way to highlight the benefits of SC//HyperCore is through real world scenarios at the various levels of the SC//HyperCore cluster: cluster formation (installation), networking, migration, software and hardware. The following sections will describe how SC//HyperCore intelligence simplifies virtualization and how SCRIBE capabilities unify storage management from the initial setup and configuration of the SC//HyperCore cluster through day-to-day operations, hardware failures and replacements and eventual capacity and performance expansions.

Cluster Formation—Resource Aggregation

Once the SC//HyperCore nodes are racked, cabled, and configured with physical network connectivity, the initialization process takes multiple nodes and logically bonds them together to act as a single coordinated cluster. Once the cluster is formed, it is managed as a single system and all resources attached to each node are aggregated into a single managed pool of storage and compute resources. This is real hyperconvergence.

SC//HyperCore management architecture eliminates the need for a separate management server (or VM appliance) to install or manage and no single "brain" or "master node" (known as a single point of failure) that controls the overall system. The entire SC//HyperCore cluster is managed by pointing a web browser to the public IP address of any node in the cluster for a complete cluster management view utilizing a simple and intuitive web interface.

Networking—High Availability

While each node in the cluster has its own physical network identity and interfaces, the ability to move and place virtual machines on specific nodes lets SC//HyperCore intelligently balance compute and I/O load among nodes and network interfaces in the cluster, maximizing overall system performance.

SC//HyperCore nodes have two distinct physical networks in which they participate—a public LAN network and a private Backplane network. All current SC//HyperCore nodes offer redundant network ports for both the LAN and Backplane network connections to allow for full network redundancy. The LAN network provides a path to allow network access to the management interface, virtual machines and end users accessing data from VM's running on the cluster. The private Backplane connection is for intra-cluster communication, such as the creation of redundant copies of data being written on one node or disk to a mirrored copy on another node.

Migration—Move Existing Workloads to SC//HyperCore

If the customer has existing workloads that they want to move to SC//HyperCore, there are a variety of Scale Computing or third-party tools available that can migrate existing physical server (Physical-to-virtual— P2V) and virtual machine (Virtual-to-Virtual—V2V) images to SC//HyperCore VMs. Tools like Acronis Cyber Cloud even support "convert to VM" functionality, allowing you to backup a VMware or Hyper-V based VM and convert it to a VM running on SC//HyperCore directly.

SC//HyperCore Move Powered by Double-Take™ is available from Scale Computing and offers near-zero downtime migration of Windows servers and applications onto SC//HyperCore. Several other tools have been outlined in Application Notes that can be found on the Scale Computing Customer Portal. In addition, Scale Computing and its partners offer full or quick start migration services to make it easy for customers to begin using SC//HyperCore.

Management – Multi-Site Management

HyperCore UI allows for monitoring and management of multiple clusters across sites from a single console. Multiple clusters across sites can be added to the same console view for monitoring and then as needed, launch into a full system management interface for each individual cluster. Multi-site management allows dozens or hundreds of sites and clusters to be monitored and managed quickly and efficiently.

Management – Role-based Access Control

Administrator roles can be managed within SC//HyperCore by assigning specific sets of access controls to individual administrators. As a new administrator account is created, access to specific administrator functions can be granted or restricted to provide only the level of access needed. Access controls are logically grouped to include role-based functions such as VM creation, VM deletion, backup/disaster recovery, and more.

Software—Real-Time System Monitoring

Utilizing a management state machine at the hypervisor layer, all nodes in the SC//HyperCore cluster work in tandem for real-time health and state monitoring—essentially creating a self-healing cluster. All nodes, hardware components, software components, and guest VMs are monitored in real-time. The cluster will automatically take action on any identified conditions and notify configured technical contacts through email of any issues, again in real-time. Contacts can then take action or contact Scale Computing Support if needed—often, the SC//HyperCore system is intelligent enough to resolve issues without manual intervention ever being required.

Additionally, HyperCore UI will display active cluster conditions in real-time. Once again, these conditions are persistent across all nodes as the cluster works as a single system, so it is not required to access the management interface through the identified node LAN IP in order to view or take action on an issue.

Software—Rolling Upgrades

SC//HyperCore offers automatic, non-disruptive firmware upgrades across the entire SC//HyperCore cluster. The HyperCore UI detects and displays available updates that can be initiated by the administrator. During the automated update process, SC//HyperCore live-migrates running VMs from a node, updates the node firmware, returns VMs (through live migration) to their original node location, and repeats this process for each node until the entire cluster is updated.

SC//HyperCore updates require sufficient compute RAM available to allow the VMs running on each node to be migrated across the remaining nodes in the cluster (essentially, the ability for all VMs to run with one node's RAM resources unavailable in the cluster). This is also a requirement to allow for proper VM failover and SC//HyperCore continuously monitors used versus available RAM and generates an alert condition through the statemachine if there is not sufficient memory to allow VMs from any node to failover to remaining nodes in the cluster.

Software—Cluster-to-Cluster Replication

SC//HyperCore cluster-to-cluster replication provides the ability to replicate VMs—at a per VM level—on one SC//HyperCore cluster another, often at a remote location. Cluster-to-cluster replication is designed to run continuously and to transmit changes to a secondary cluster as quickly as possible. SC//HyperCore VM replication leverages the SCRIBE snapshot capabilities plus additional compression functionality to efficiently determine the data changes between the most recent data snapshot and the snapshot that represents the last completed replication point (often the last completed point is just minutes old) and transmit the changes to the remote cluster.

Leveraging change tracking limits the impact of replication on the cluster by reducing the I/O required to read and transmit changes. This also eliminates the need to "brute force" read and compare data to determine which blocks have changed since the last replication cycle.

For VMs that were created from snapshots or via cloning that share common data blocks (such as multiple VMs created from a "template" and cloned), SC//HyperCore is intelligent enough to transmit those common blocks across the network to the remote cluster a single time which can greatly reduce the bandwidth required for the initial replication of a new VM created from an already replicated template.

Hardware—Disk Failure Scenario

SC//HyperCore was designed with the expectation that hardware eventually fails, and such failures should not be critical to the system. Based on this design, disk failures are considered trivial and have little effect on the cluster beyond temporary loss of the capacity and I/O resources of the affected disk. When a disk is determined to have failed, in addition to raising alerts in the management interface and through email, SC//HyperCore will continue servicing I/O transparently using the mirrored data copies. Additionally, SC//HyperCore will automatically use remaining available space to re-generate a second copy of any chunk that was previously located on the failed disk. Any new writes are mirrored across the remaining disks to ensure immediate protection of new and changed data and reduce future data movement once the failed disk is replaced.

Unlike many RAID systems, disk failure and replacement does NOT require time consuming and I/O intensive parity calculations to determine what data was lost. SC//HyperCore simply reads the mirror blocks and writes a new copy using any available space on the cluster. Note that both read and write operations here leverage the same wide striping performance benefits as normal I/O.

Not only does SC//HyperCore not require a dedicated "hot spare" disk that sits idle most of the time, it doesn't require a failed disk to be replaced or any other manual steps before it automatically re-creates the lost mirror copies to regain full data redundancy. This ensures there is no critical need to immediately replace a failed disk in the system for ease of system management.

Hardware—Node Failure Scenario
SC//HyperCore was designed to not only to be resilient against disk failures, but to sustain the temporary loss of an entire node while continuing to process data and running VMs using redundant resources from the remaining cluster nodes. A single node failure (loss of access to all node disks—SSD or spinning disks— and compute resources) does not impact data availability for VMs, unlike some systems on the market where even a single SSD or spinning disk failure may result in data loss due to caching or RAID arrays.

Hardware—Network Failure Scenario
All current SC//HyperCore nodes offer redundant network ports for both the public LAN and private backplane network connections for a total of 4 network ports per node to allow for full network redundancy. These ports are in an active/passive bond for failover—2 ports bonded for LAN access and 2 ports bonded for backplane access. If access to the primary port is lost on either the public or private node network the secondary bonded port will activate with little to no disruption to the cluster node. Once the primary port becomes available again the network will fail back to that port. All onboard network cards are used as the primary port for the SC//HyperCore nodes.

Hardware—Expanding the Cluster
After the initial cluster installation and formation, SC//HyperCore allows additional nodes to be added to the cluster at any time with no downtime to the system. Joining a new node to the cluster automatically adds the new node's capabilities to the overall available storage pool and compute capacity. New nodes can be of the same or different type or capacity as the existing nodes in the cluster. SC//HyperCore allows newer, faster nodes with different CPU, RAM and disk configurations to be added to existing SC//HyperCore clusters to increase capacity and to ultimately replace and retire older legacy nodes as needed.

When adding nodes with storage resources to an existing cluster, existing storage may be optionally rebalanced to immediately utilize the new nodes and free up resources on older nodes. This process will relocate blocks from the older nodes onto the newer nodes to achieve an equal distribution of data across cluster nodes and disks. Even without using the option to rebalance data, SCRIBE will prioritize the node with the most free storage capacity for any new data written to the cluster, so the new node storage resources will immediately become an active part of the cluster. With or without rebalance, the new resources will begin providing more distributed I/O and improved system performance across the cluster.

Running SC//HyperCore VMs may be immediately live-migrated to the newly added nodes as needed to take advantage of the new compute resources with no downtime or disruption to VMs. New CPUs and RAM, in addition to storage resources, can be utilized by both VMs that are live migrated or new VMs that are created within the SC//HyperCore cluster.

Resources

Additional informational and technical resources are available from Scale Computing at www.scalecomputing.com/resources.

Provide Feedback or Contact Support
Any comments or suggestions regarding this document or other Scale Computing documentation, you can send them to documentation@scalecomputing.com. If you need help, call +1-877-722-5359. You can also email Scale Computing Technical Support at support@scalecomputing.com or find us on the web at www.scalecomputing.com.