HC3®, SCRIBE and HyperCore™

Theory of Operations

This document is intended to describe the technology, concepts and operating theory behind the Scale Computing HC3 System (Hyper-converged Compute Cluster) and the HyperCore™ OS that powers it. It also covers the new storage layer SCRIBE introduced with HyperCore. It is assumed that the reader is already familiar with the basic virtualized compute features and benefits of HyperCore-based clusters and high-level components of an HC3 cluster (nodes, drives, network switches, etc.).

Design Goals....................2
HC3, SCRIBE and HyperCore™ Overview....................2
  Software Defined Storage....................2
  Software Managed Compute....................2
HC3 and HyperCore In Action....................2
  Cluster Formation - Storage and Compute Resource Aggregation....................3
  Create a Virtual Machine....................3
  HC3 Virtual Machine Placement....................4
  HC3Protect™ Intra Cluster Mirroring and Wide Striping....................4
  Expanding the Cluster....................5
  Disk Failure Scenario....................5
  Node Failure Scenario....................6
    Virtual Machine Failover....................6
    HyperCore Rolling Upgrades (Planned Node Maintenance)....................6
    Storage Pool Verification / Orphan Block Detection....................6
  HyperCore VM Level Snapshots....................6
    VM Promote / Revert....................7
    VM Snapshot Based Thin Cloning....................7
  Cluster-to-Cluster Replication....................7
  VM Replication and Recovery....................8

Design Goals

Scale Computing's HC3 and the HyperCore architecture were designed to provide highly available, scalable compute and storage services while maintaining operational simplicity through highly intelligent software automation and architecture simplification. HyperCore puts intelligence and automation in the software layer and was designed to take advantage of low cost, easily replaceable and upgradable "commodity" hardware components including the virtualization capabilities built into modern CPU architectures. By clustering these components together into a single unified and redundant system, these attributes combine to create a flexible and complete "datacenter in a box" that operates as a redundant and elastic "private cloud" with additional nodes being automatically "incorporated" into the cluster and failed hardware being expected, and easily replaced with minimal effort or disruption.

HC3, SCRIBE and HyperCore Overview

HC3 and HyperCore are based on a 64-bit hardened and proven OS kernel – and leverage a mixture of patented proprietary and adapted open source components. As a result, Scale Computing has full control over the entire hardware and software stack enabling Scale to optimize and enhance all software layers for simplicity and reliability and tune the software for optimal performance.

Software Defined Storage

A critical software component of HyperCore is the Scale Computing Reliable Independent Block Engine, known as SCRIBE, an enterprise class clustered block storage layer, purpose built to be consumed by the HC3 embedded KVM based hypervisor directly. SCRIBE discovers and aggregates all block storage devices across all nodes of the system into a single managed pool of storage. All data written to this pool is immediately available for read or write by any and every node in the storage cluster, allowing for sophisticated data redundancy and load balancing schemes to be used by higher layers of the stack such as the HC3 compute layer. More unique aspects of SCRIBE will be detailed later in this paper.

Software Managed Compute

The HyperCore compute software layer is a lightweight, type 1 (bare metal) hypervisor that is directly integrated into the OS kernel, and leverages the virtualization offload capabilities provided by modern CPU architectures. Specifically, it 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. HyperCore integrates the SCRIBE block storage objects directly into the KVM hypervisor. This means that virtual machines running on HC3 have direct access to SCRIBE "virtual storage device" (VSD) objects in the clustered storage pool without the complexity or performance overhead introduced by using remote storage protocols and accessing remote storage over a network.

Unlike other seemingly "converged" architectures in the market, the storage layer is *not* a Virtual Storage Appliance or VSA but instead interfaces directly with the hypervisor allowing data flows to benefit from zero copy shared memory performance. Nor is SCRIBE simply 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. Performance killing issues such as disk "partition alignment[1]" with external RAID arrays go away. 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 a thing of the past.

HC3 and HyperCore In Action

One of the best ways to highlight the benefits of the HC3 design is to follow it in action through a typical real world usage scenario. The following section will describe how HyperCore intelligence simplifies virtualization and storage management from initial setup and configuration, through eventual capacity and performance expansions, hardware failures and replacements.

[1] 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 different block size which itself ultimately will sit on top of some other physical or logical block storage that may have it's own block size and physical data layout.

Cluster Formation - Storage and Compute Resource Aggregation

Once the Scale Cluster nodes are racked, cabled and configured with physical network connectivity, the cluster formation process takes multiple nodes (currently a minimum of 3) and logically bonds them together to act as a single coordinated system. Once a 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.

With HC3 there is no separate management server (or VM appliance) to install or connect to or a single “brain” that controls the overall system. You can manage the entire system simply by pointing a web browser to the LAN IP address of any node in the cluster and you will get the same cluster-wide management view. Also, all nodes of the HC3 system coordinate with each other to monitor the overall state of the cluster, nodes, hardware components and virtual machines running across the entire system.

While each node has its own physical network identity and interfaces, the ability to move and place virtual machines on specific nodes lets the HC3 intelligently balance compute and I/O load among nodes and network interfaces in the cluster, maximizing overall system performance.

HC3 nodes have two distinct physical networks in which they participate[2]. A public network provides a path to allow network access to virtual machines, as well as for and users accessing data from VM’s running on the cluster. A 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.

For more information on the mirroring method used in HyperCore, please see the section titled HC3Protect™ Intra Cluster Mirroring and Wide Striping.

After forming an HC3 cluster, the next step is to create your first virtual machine. There is no need to provision storage for use by VM’s, create iSCSI luns or NFS shares, or create a “datastore” for VMs use by connecting the hypervisor to that storage. None of those concepts even apply to a hyperconverged HC3 system.

Create a Virtual Machine

Creating a virtual machine is a simple process that is accomplished through the Scale Computing HC3 Manager web browser user interface. Selecting the Create option from the Virtualization tab allows the user to specify required and optional parameters for the virtual machine including:

- Number of virtual CPU cores
- RAM
- Number and size of virtual disks to create
- Attach or upload a virtual DVD/CD ISO image for installing an operating system

Creating a virtual machine not only persists those VM configuration parameters that will later tell the hypervisor how to create the virtual machine container when it is started, it also physically creates storage objects using the SCRIBE distributed storage pool that will contain the virtual hard disks to present to the VM once it is started. HC3 Virtual Machines are able to access their virtual hard disk files directly as if they are local disks, without the use of any SAN or NAS protocols, regardless of where they are running at the time.

HC3 virtual disks are thin provisioned so that virtual hard drive space is not fully allocated until it is actually used by the guest virtual machine.

[2] All current HC3 nodes offer redundant network ports for both the public LAN and private backplane network connections to allow for full network redundancy.

The HC3 web UI is used to upload Operating System and Virtual Appliance ISO installation images for virtual machine installation as part of the VM creation process (ISO's are virtual DVD / CD images). Once uploaded, ISO images are available for use by future VMs.

Migrating Existing Workloads to HC3

If the customer has existing workloads that they want to move to HC3, there are a variety of Scale Computing provided or third-party tools available that can migrate existing physical server (P2V – Physical-to-virtual) and virtual machine (V2V – Virtual-to-Virtual) images to HC3 VMs.

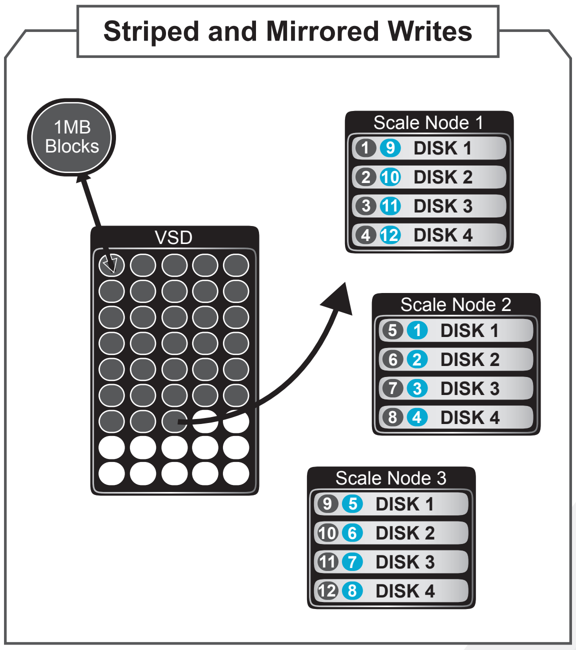
HC3 Move Powered by Double-Take is available from Scale Computing and offers near-zero downtime migration of Windows servers and applications onto HC3. Several other tools have been outlined in Application Notes that can be found on the MyScale portal. In addition, Scale Computing and its partners offer full or quick start migration services to make it easy for customers to begin using HC3.

HC3 Virtual Machine Placement

Because all HC3 capable nodes have access to the entire pool of storage, any virtual machine can be started on any HC3 capable node of the cluster based on the availability of compute resources that VM requires. For example, if a new VM requires 16GB RAM there may only be certain nodes with that much currently available and HC3 makes this node selection automatically. HC3 allows running VMs to be "live migrated" to other HC3 nodes without the VM being shutdown and with virtually no noticeable impact to the workload being run or clients connecting to it. In addition, should an HC3 node fail, any VMs that were running on that node will be quickly re-started on remaining nodes since every HC3 node has access to the same pool of redundant storage.

HC3Protect™ Intra Cluster Mirroring and Wide Striping

While HyperCore treats all storage in the cluster as a single logical pool for management and scalability purposes, the real magic comes in how data blocks are stored redundantly across the cluster to maximize availability as well as performance.



The SCRIBE Storage Layer allocates storage for block I/O requests in chunks ranging from as large as 1MB to as small as a 512 byte disk sector. SCRIBE ensures that two or more[3] copies of every chunk are written to the storage pool in a manner that not only creates the required level of redundancy (equivalent of 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. 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.

Take the example of 1GB of data being written to a cluster of 3 of Scale Computing's smallest HC100 nodes, which contain a total of 12 hard drives (each node contains 4 drives with 500GB RAW capacity, thus contributing 1TB of effective storage space to the cluster after accounting for 2 copies of all data blocks being stored). That 1GB of data (or 1024MB) would require 1024 chunks (1MB each) to store the first copy plus 1024 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 170 x 1MB writes in order to store that 1GB redundantly.

[3] In the current release, all data is replicated two times for redundancy. In a future release the number of copies will be selectable at the VM level allowing greater replication across a larger number of nodes if desired.

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 drives in the cluster, and 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 as well.

Contrast this 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 drives 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 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).

Later in this document we will discuss various hardware failure and replacement scenarios to show how HyperCore leverages the redundancy provided by these mirrored blocks in various real world scenarios.

Expanding the Cluster

After the initial cluster formation, HyperCore was designed to allow additional nodes to be added to the cluster at any time. Joining a new node to the cluster automatically adds its storage to the storage pool increasing storage and compute capacity and I/O handing. If the nodes are capable of running virtual machines (verses simply expanding storage resources), they are added to the HC3 managed compute pool and available for new VMs or for existing VMs to be live migrated to distribute load. New nodes can be of the same or different type or capacity as the existing nodes in the cluster. HC3 was designed to allow newer, faster nodes with different CPU, RAM and disk configurations to be added to existing HC3 systems over time.

When adding nodes with storage resources to an existing cluster, while not required, HyperCore does provide the capability to re-balance existing storage to utilize all the new resources of the cluster. This process will essentially move some blocks from the older nodes onto the newer nodes to achieve an equal distribution of data across cluster nodes and drives. Consult with Scale Computing technical support to determine if this is recommended based on the current utilization of the cluster. For HC3 Virtual Machines, it is simple to immediately “move” live running virtual machines to the new node to take advantage of the new compute resources with no downtime or disruption.

Disk Failure Scenario

HyperCore was designed such that disk failures have little effect on the cluster beyond the temporary loss of the capacity and I/O resources of that disk. When a disk is determined to have failed, in addition to raising corresponding alerts in the Scale Computing Cluster Manager UI and remote alerting mechanisms, HyperCore will continue servicing I/O transparently using the mirrored data copies. In addition, 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.

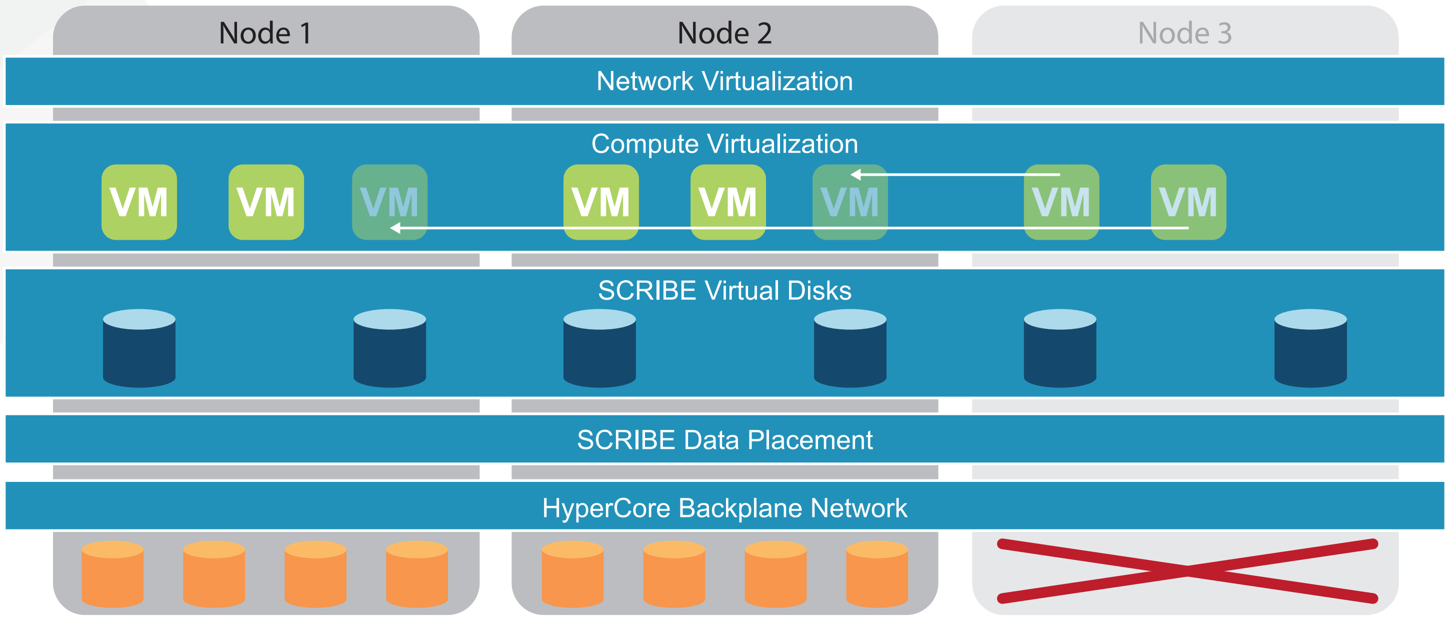
Note that unlike many RAID systems, this process does NOT need to perform time consuming and I/O intensive parity calculations to determine what data was lost. HyperCore simply reads the mirror blocks and writes a new copy using any available space on the cluster. Note that both read and write operations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

Not only does HyperCore not require a dedicated “hot spare” disk that sits idle most of the time, HyperCore does not even require a failed disk to be replaced or any manual steps before it automatically re-creates the lost mirror copies to regain full data redundancy.

Node Failure Scenario

HyperCore was designed 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. Because of the number of different scenarios possible, we recommend that customers work with our support team on the best plan of action if the node failure is considered permanent.

Virtual Machine Failover
If an HC3 node was running VMs prior to failure, those VM disks and data are still available to the remaining cluster nodes. This allows VMs to be automatically re-started on the remaining nodes with HyperCore selecting the ideal node placement for restarting.



HyperCore Rolling Upgrades
HyperCore added the ability to do automatic non-disruptive upgrades of itself across the entire cluster. When an update is shown as available in the UI and the upgrade process initiated, HC3 will automatically live migrate running VM's from each node, update the OS kernel, hypervisor (KVM), storage (SCRIBE) and management software in a single automated step and then return VM's to their original location using live migration when the upgrade is complete. The only requirement is that there is sufficient compute RAM available to allow the VMs running on each node to be migrated across the remaining nodes in the cluster. Since this is also a requirement to allow for VM failover, HC3 continuously monitors used vs. available memory and generates an alert if there is not sufficient memory to allow VM's from any node to failover to remaining nodes in the cluster.

Storage Pool Verification / Orphan Block Detection
For temporary failures such as a node power outage, HyperCore will automatically verify that any data blocks it contains were not changed elsewhere on the cluster while the node was off-line and will free the original blocks to the pool if they are no longer required (for example by a snapshot). For example, it is possible that block 200 was contained on node 1 and 2 and while node 1 was down due to a power outage, block 200 had changed and would have been updated on node 2, plus a new redundant copy written elsewhere on the cluster. When node 1 is back online, HyperCore will detect that the original contents of block 200 are no longer current or needed and correct that discrepancy.

HyperCore VM Level Snapshots
HyperCore offers the ability to take multiple point-in-time consistent snapshots of virtual machine storage objects (virtual disks) and their VM configuration. Snapshots occur nearly instantly with virtually no impact on production workloads. 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. 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 have 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

Deleting a snapshot is simply a matter of updating the reference counts for blocks referenced by that snapshot and returning any that go to 0 to the capacity pool.

This 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 Promote / Revert

Any HC3 VM level snapshot can be promoted to a live VM and started nearly instantaneously from the HC3 management UI by leveraging the “clone” action. If the goal is to revert the running VM to a previous point in time, for example to undo a problem patch, you can simultaneously clone the previous snapshot to a live VM and start it while you stop and optionally delete the problem VM. Or, retain the problem VM image and start it isolated from the product network to do a post mortem analysis. When promoting or reverting a VM, the entire VM configuration and it’s snapshotted storage objects are made available allowing you to edit the VM configuration as you chose 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 also allow for instant space efficient cloning or the creation of “template” VM’s. In a single step, you can clone a live running VM by taking a snapshot and immediately promoting the snapshot to live, starting it in an isolated network and doing a trial run of a new software patch for example. Golden Master template VM’s can be created with the latest OS and application patches, security and AV tools and settings and used to quickly deploy large numbers of similar server or desktop OS VM’s. Each VM created from a snapshot clone originally takes zero additional space and grows based only on the changes made once that clone is booted.

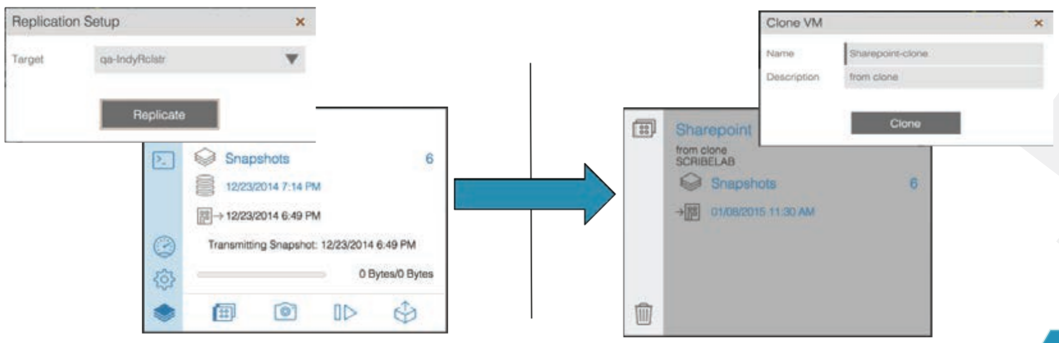
Unlike many snapshot and cloning architectures, there is no dependency on a parent VM or hierarchy of snapshots. You can delete the parent VM/template without impacting any VM’s cloned from it. You can clone a VM snapshot and keep the original VM running and changing blocks or even delete it without impacting or corrupting any clones.

Cluster-to-Cluster Replication

HyperCore cluster-to-cluster replication provides the ability to replicate selected virtual machines on one HC3 cluster to a second cluster, often at a remote location. Cluster-to-cluster replication is designed to run continuously to transmit changes to a secondary cluster as quickly as possible. HC3 VM replication leverages the SCRIBE snapshot capabilities plus additional functionality to efficiently determine the data changes between the “current” data snapshot and the snapshot that represents the last completed replication point which may be just minutes old.

Leveraging effecient change tracking reduces 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 virtual machines that were created from snapshots or via cloning that share common data blocks (such as multiple VM’s created from a “template” and cloned), HC3 is intelligent enough to transmit those common blocks across the network to the remote cluster only once which can greatly reduce the bandwidth required for the initial replication of a new virtual machine created from an already replicated template.



Target: qa-IndyRclstr

12/23/2014 7:14 PM

→ 12/23/2014 6:49 PM

Transmitting Snapshot: 12/23/2014 6:49 PM

0 Bytes/0 Bytes

Name: Sharepoint-clone

Description: from clone

from clone
SCRIBELAB

→ 01/08/2015 11:30 AM

VM Replication and Recovery

HC3 replicated VM’s as well as their snapshots can be manually activated and made live with a click of a button. The recovery process from a remote replicated cluster is the same as recovering from a snapshot on the original cluster.

The entire process can later be reversed to replicate and return one or more VMs back to the original cluster. As with the original replication, HC3 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.

Also note that for immediate and automatic failover at the VM level between HC3 and non HC3 workloads – optional HC3 Availability powered by Double-Take™ replication and failover software can be run inside the guest operating systems on non HC3 platforms replicating to or from VM’s running on HC3 target systems. This approach could also allow HC3 to be used as an off-site recovery and failover target for workloads running on physical machines or other virtualization platforms.

Corporate Headquarters
5225 Exploration Drive
Indianapolis, IN 46241
P. +1 317-856-9959

West Coast Office
San Francisco, CA

EMEA Office
Saunders House
52-53 The Mall
London
W5 3TA
United Kingdom

www.scalecomputing.com

1-877-SCALE-59 (877-722-5359)