SCALE COMPUTING

Alternative Architectural Approaches of Distributed Edge Locations

From the Experts at Scale Computing

Technical White Paper

Abstract

This paper will present alternative architectural approaches to support the IT needs of distributed edge locations from traditional ROBO (remote office/branch office) use cases to modern, hybrid, cloud-native applications. The paper will examine the costs and operational pros and cons of architectures based on fortified, enterprise-class servers vs. two-node standby server pairs/clusters vs. integrated autonomous hyperconverged systems such as Scale Computing Platform.

Table of Contents

Introduction

IT at the Distributed Edge of your Network

Your enterprise is an ever-evolving mix of application workloads that are required to support operations across dispersed locations that may number in the tens, hundreds or thousands. If you are like most, you are responsible for maintaining legacy applications (Windows/Linux and VMs), as well as the current and/or future need to support new, modern application architectures (such as containers/cloud native apps/IoT/analytics). These applications need to be distributed to the edge of your business, close to where data is generated, analyzed, and used for real-time decisions and actions. Many of these applications generate enormous volumes of data and require state of the art compute processing resources capable of maintaining low-latency, autonomous operation that is not dependent on remote network connectivity.

Cost is a primary factor. In this multi-site, distributed edge computing environment – even minor differences in CapEx and OpEx costs per location will multiply and massively increase total cost of ownership. Further - it's important to look beyond initial hardware, software acquisition costs and initial deployment to consider the cost of ongoing operational support for these dispersed locations (monitoring, remote troubleshooting and especially the need for on-site visits) as well as the impact of application downtime and recovery objectives given that things inevitably fail at the worst times, and in the worst locations.

Unless this is a brand new set of applications, you probably have a single server (maybe a glorified desktop machine), perhaps two servers, or even more already deployed in each location. And you found a place to put them, power them, cool them, and secure them (physically and logically) for better or worse. You've cobbled together ways to monitor, update (often too late), and repair them - or pay third parties large service fees to perform some or all of those functions with "acceptable" SLAs and accepting the risk of inevitable downtime until properly trained help (and likely replacement parts) can arrive at the remote site.

Many times, the remote office scenario is even less ideal, with centralized IT staff getting surprise calls (often outside working hours) from panicked application users or remote managers complaining that their application(s) are down and figuring out, on the fly, what to do: how to get someone there, scrambling to locate spare parts, hoping there are good (tested) backups available and that maybe this downtime will be measured in hours, not days or worse. The costs of downtime like this in lost business and productivity are often hard to calculate until they have been experienced and then the blame game begins.

This paper will walk through a range of infrastructure redundancy options available to remotely support the IT needs of distributed edge computing architecture and present the associated costs, risks, and benefit tradeoffs for each.

Status Quo

Simplicity of the Monolithic Remote Office Server

Something could be said for the simplicity of older monolithic systems where one app runs on one OS on one single server with its own local storage. The rise of virtualization even made it possible to run a few applications on that monolithic server, as long as you were willing to accept **all** of those applications going down in the event of various failures. It was a simple architecture where you hope failures don't happen very often or at the worst time, do regular backups and hopefully test recovery for when failures do happen. Essentially "accept" a certain level of downtime, data loss, lost productivity and potentially lost business and customer satisfaction issues that can result.

But the bar has been raised with customer expectations and with internal operational dependence on these systems. Downtime is no longer considered normal or acceptable and the impacts of a downtime event can extend far beyond the immediate event. Beyond that is the need to allow regularly scheduled planned downtime for OS, application, and hardware updates and maintenance.

Server Fortification

The more important those applications in total are (cost of downtime, lost productivity, etc), the more likely you will spend money at every remote location to alleviate the most frequent and expected single points of failure (SPOF) - at extra cost. Disks and storage are a common SPOF (as well as potential performance bottleneck), so it is very common failure mitigation to add costly RAID storage controllers and redundant disks to ensure storage is more highly available, performant, and that data is less likely to be lost completely with a disk failure.

Server power requirements, including redundant power supplies and UPS's are also very common investment intended to prevent downtime. Similarly, using multiple network connections and switches is common for redundancy and load balancing to prevent the network from being a single point of failure or performance bottleneck.

When something goes wrong with a monolithic remote server, applications are down, so you know it fairly soon, and can start troubleshooting various aspects of that box, software, network, etc. That means having someone sufficiently IT savvy and familiar with your unique setup go to that box and start digging in, eliminating possible causes, likely by going on site ASAP, hopefully with the right spare parts to repair or replace the server, possibly restore data and applications and **eventually** the site is back in business. But there are many components like motherboard / CPU / RAM / RAID cards that can't easily be fortified and remain single points of failure.

Further, all of this hardening is just to provide hardware component redundancy within a single box, and does nothing about providing, or **maintaining** true, independent redundancy for your valuable data. This is why periodic backup and restore[1] is absolutely critical for when, not if, they are needed.

[1] Beyond the scope of this paper -- it's also important to consider full recovery time requirements for applications – including the entire OS, application software, and data, as well as historical retention policies to allow roll back in the event of undesirable changes such as ransomware, failed OS, or application updates or data corruption. SC//Platform offers scheduled snapshot retention, remote snapshot replication to centralized data center or cloud as well as third party backup software integrations to comprehensively address these needs across edge locations.

Eliminating **All** Single Points of Failure

Whether you've been burned by downtime for a critical application, or are just wisely thinking ahead and planning for the inevitable – you might have considered getting a complete duplicate system to eliminate **all** single points of failure. Buy two of everything, leverage application specific mirroring / failover logic (like load-balanced web servers or redundant video management servers for archiving) or general OS or storage level mirroring and failover solutions to synchronize an active primary server with a passive standby server. If done correctly, this can be a viable solution, however, you will generally **more than double** your costs for resources that sit there idle / passive and you hope you never need to use. And many "bolt on" systems or add on features impose significant performance overhead on your applications… requiring you to buy **even bigger systems - times two** - to get the same amount of work done.

Application Availability vs. Data Redundancy

Redundant server pairs can keep an application running, but after a failure you are now back to the state of that original single server, **now** fully responsible for continued operation and storage. When you lost your primary system and switched to your backup system, **you lost your backup plan and redundant storage location**, so getting the primary system back online quickly is still a top priority and time critical.

For any new or changed data, there is only one server up now – so there is no secondary server available to mirror data to. You may not be experiencing downtime at the moment, but your data is now vulnerable and one failure away from being completely lost until the primary system is restored AND a potentially significant resync of data changes from backup back to the primary is complete (often impacting application performance.)

As a result of this loss of data redundancy – most of these systems require or recommend using **both** local RAID within each server + remote mirroring of all data to the second system resulting in "usable" storage capacities often less than 25% of total raw capacity across both systems.[2]

[2] If each of two servers contains 4 x 2TB SSD's for total of 16TB raw storage. If each server first created a local RAID array (either RAID6 or Raid5 with spare) from the disks, that would result in 4TB usable storage. Which would be mirrored to the RAID array on the second system. So ultimately that system can store 4TB of data - or 25% of the raw 16TB storage.

Is there a better way?

Yes. A better way is to design the system to make normal failures normal, even expected. Design for the system and applications to be healthy and protected even with a disk failure, or a whole node down. Assume one thing (disk, node, or other component) may always be down, and when that's not the case things are even better.

Redundant Array of Disks Across Redundant Servers

Storage, whether on spinning disks or solid state, is one of the most likely and most critical failures to protect against. Failure of spinning disks is well known and common and is one of the reasons solid state storage is replacing spinning rusty disks as a storage media wherever practical. Solid state drives (SSDs) provides increased reliability with no moving parts that are susceptible to vibration and with parts that are less susceptible to heat, etc. But even SSDs are consumable and do wear out over time, and of course they aren't available if the compute host they are installed in is itself down or crashed (or stolen, or attacked by ransomware).

In the single monolithic server example above, it was mentioned that RAID is commonly required to protect data availability within a single server (as well as provide performance benefits vs. individual disks.) What if the concept of RAID was extended to provide redundancy using disks located in different servers? Now that same storage capacity overhead required for RAID could not only protect against failure of the storage media, but also against failure or temporary downtime of the server hosting that storage. This is the general concept of RAID but with data redundantly distributed across disks located in **different servers, allowing the same redundant copy to protect against either disk or server failure**.

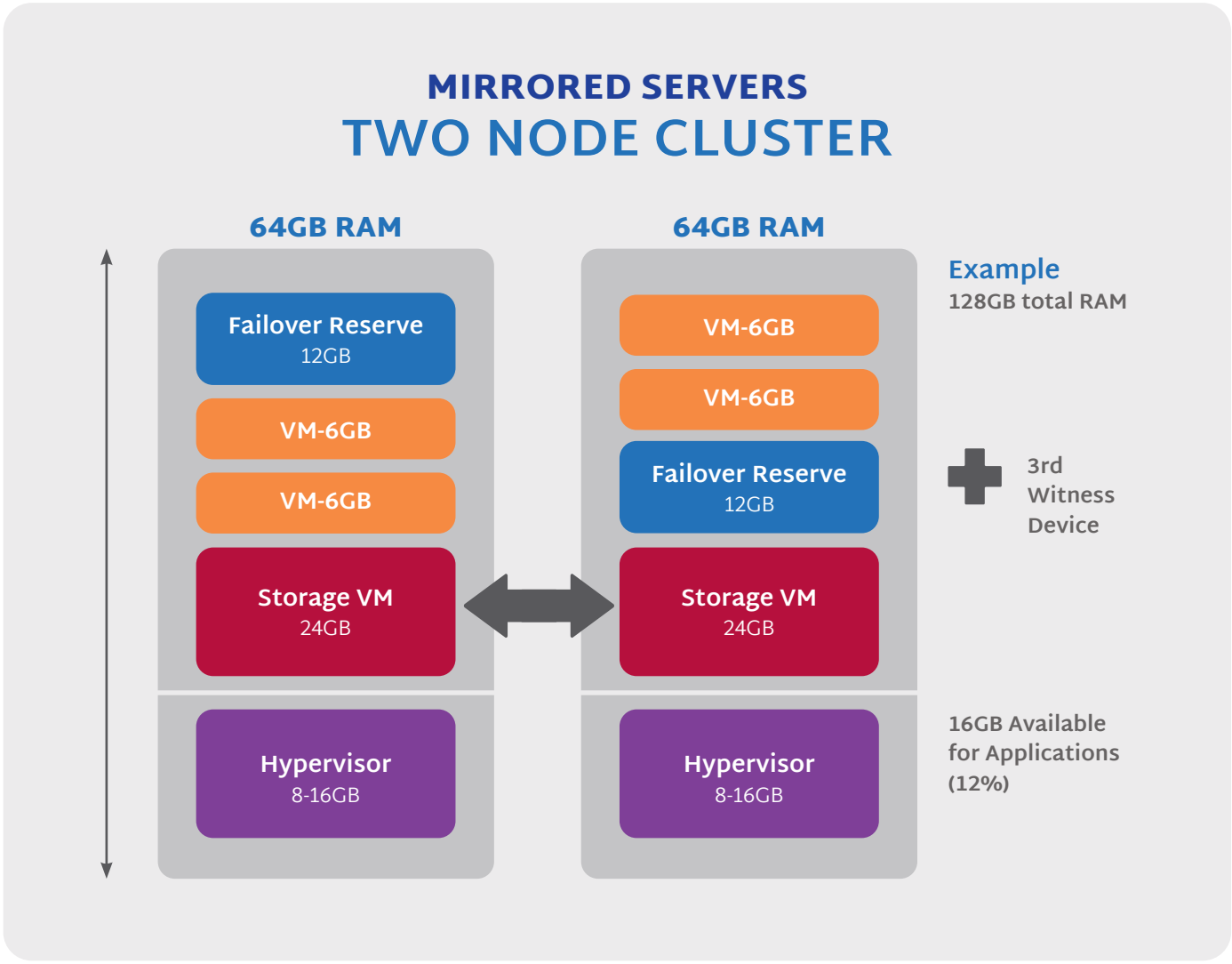
Aren't Servers Expensive Though?

Some servers are **far** more expensive than others. If you've already priced out a powerful and resilient enterprise-class server (often paying more for a system with empty slots/sockets to allow for future scale up expansion), buying a duplicate second server that just sits idle and ready to take over in case of failure **more than doubles your cost** at every location and gives you more to monitor and manage.

You could run active/active configurations where both servers do "some" work all the time – but take over the others workloads in the event of a failure. But even in that case, ½ total available / usable resources of the server pair in total need to be idle / free so that each node is ready and able to take over the applications running on the other.[3] You'll still need the same server capacity whether in active/active or active/passive.

Additionally, many infrastructure solutions which offer a "two-node cluster" or mirrored server pair are already inefficient when it comes to resource usage. The operating system/hypervisor alone can consume a fair amount of resources but then to manage storage in such a cluster, often extra storage management VMs are needed that consume more resources, requiring even more server resources overall, increasing the cost. Even in cases where your intended workloads have light resource requirements, the systems you need to run them may require much bigger server than you anticipate because of the overhead needed to create the clustered redundancy.

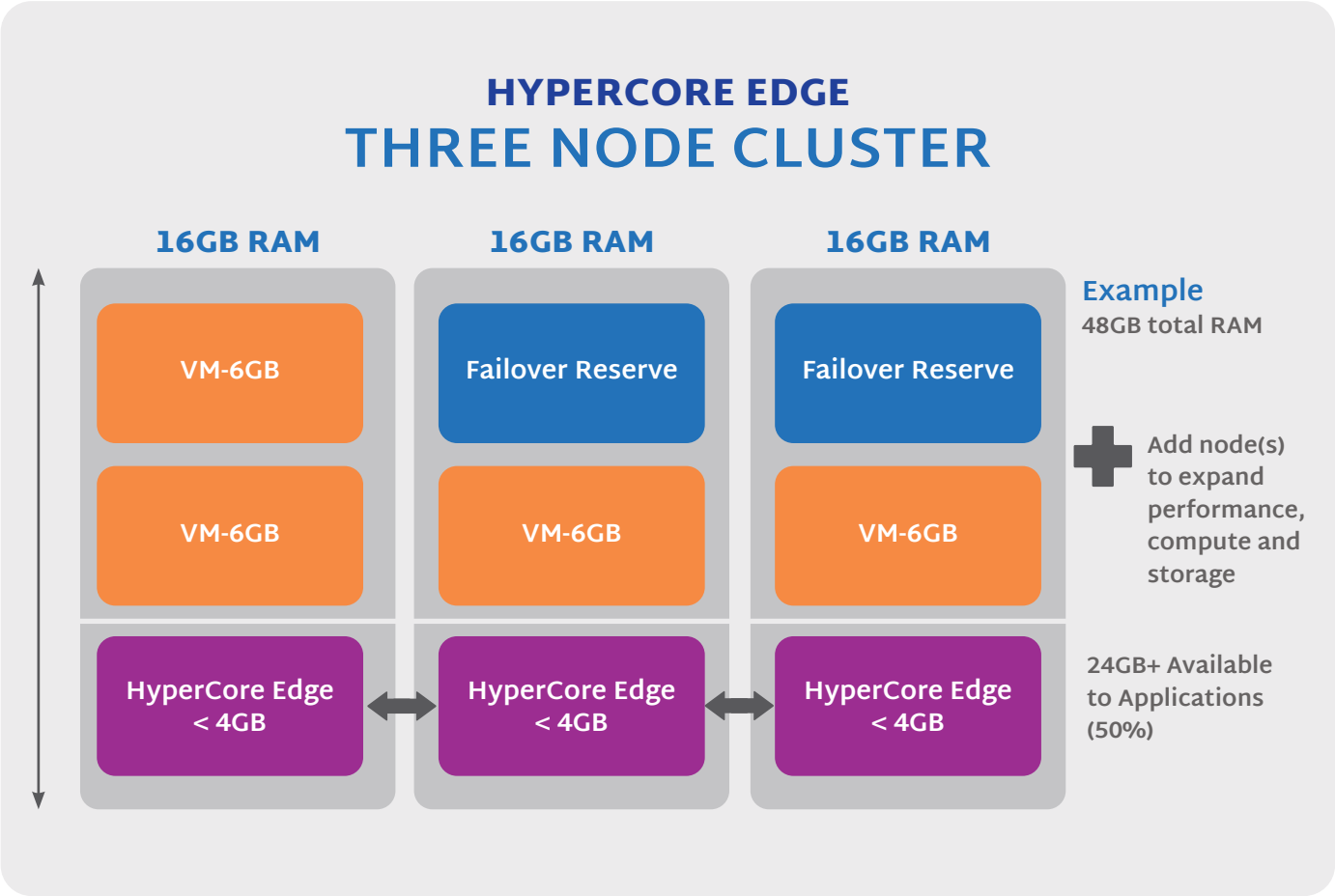
[3] In any so called "2 node cluster" or mirrored pair server system, to allow a surviving server to positively prove that a secondary server is actually down and not still running applications or modifying data, some reliable system of preventing what is known as a "split brain" cluster operation is required. This generally takes the form of some "tiebreaker" device or devices that is used to verify which servers are actually up - and which servers are actually down. If node 1 loses communication to node 2 - it can contact the tiebreaker(s) to see if the tiebreaker has visibility to node 2 or not. Generally, each node also is required to confirm "I'm alive" at some regular interval with the tiebreaker(s) else it is assumed to be down or to disconnect itself and stop running applications. In any case that "third vote" tiebreaker essentially becomes a critical 3rd part of the overall system. And in some architectures, it actually becomes more critical than the compute nodes / servers themselves.



Riddle: When Does Three Cost Less than Two, and Do More?

Is there a way around the cost of buying more than double what you need - just to have half of your total valuable resources left idle waiting to be used in the event of a failure? Yes – buy three (or more) **smaller** servers and distribute the **same compute and storage load** across those.

For the same usable compute resources and storage capacity, three smaller and less expensive servers, can be sized so that any two of the three are sufficient to run all the required applications, and provide access to all of the data. There still needs to be some excess resources available to take over in the event of a failure, but it's generally just one **smaller node** worth of compute resources, not half the entire system. So, in a three-node configuration, up to two-thirds of your compute resources can be used and fully maintained with one node down. In a four-node configuration, three-fourths can be active while still maintaining full failover capability.



More than simply providing more efficient failover capability - with three or more nodes - the system can continue to provide full, distributed data redundancy for new or changed data even with a full node down. With this added redundancy and resiliency of having both compute and data distributed across multiple servers - **the need for expensive, highly fortified, high-end servers with local RAID on each is also eliminated.** Further, the urgency of replacing or repairing the down components is significantly less, in many cases allowing the repair to be delayed for days or weeks without concern. The need to pay for expensive service contracts providing 4-hour on-site repair service and locally stocked part depots goes away as well.

The inherent resiliency benefits of a three-node or greater distributed system allows a much wider range of cost-effective and efficient systems to be used. As a result, three nodes of commodity IT gear in a cluster can not only cost significantly less than a redundant pair of high-end servers, it can even compare favorably to a larger single enterprise server when all associated costs are considered. Anyone who has shopped for IT equipment has probably noticed the natural price points and system limitations that exist in various classes of systems. These classes are often distinguished by CPU family and generation (Xeon, Xeon D/E, Core, etc.) with different maximum memory limitations from GB to TB, core count[4] and so on. Exceeding a limit and moving up from one family to the next higher to get just a little more compute capacity can often more than double the cost of the system and support.

[4] The example configurations in this paper focus on RAM and ignore application CPU core count issues for simplicity. Both can impose significant and costly sizing constraint on the machine types / classes that are suitable for each type of configuration. If these example VMs each demand 2 physical CPU cores of compute even in failover mode - the three-node cluster above could satisfy that with economical Core i5 CPU's. Running those same 4 VM's on a single server would require total of 8 physical cores which is not possible using any Core series CPU or most entry level Xeon based systems. That CPU core requirement on a single server likely would require moving up to systems that support the higher-cost Xeon Scalable Processor Series (Silver, Bronze, Gold, Platinum CPUs.)

Scale Out Hyperconverged Clusters vs. Mirrored Server Pairs

With scale-out hyperconverged systems like SC//Platform that allow additional nodes to be added seamlessly, expanding pooled compute and storage resources as needs grow, the need to buy expensive servers with unused/unpopulated internal expansion capability is negated. Only buy what you need when you need it!

Not only can you easily add new nodes with the resources you need when you need them (vs. upgrading everything in matched lockstep pairs), existing nodes can be "swapped out" for more powerful nodes easily in the future and without application downtime. Furthermore, upgrades and node additions are done live, without any downtime or reconfiguration required. The never-ending sequence of rip and replace everything "forklift upgrades" can come to an end. Older nodes can often be re-deployed for other uses if they are within their serviceable life.

Maintaining a two-node system in a mirrored pair whether active/passive or active/active requires ensuring that each independent system stays fully in sync with the other (and any tiebreaker system as previously mentioned,) in terms of software updates, configuration changes and more.

Compare that to the SC//Platform Edge system which manages software updates and configuration changes as system controlled atomic operations across an entire cluster of any number of nodes making management of the entire system as easy as managing a single server, and much easier than managing the typical pair of independent two-node failover systems.

Having three nodes allows for planned infrastructure maintenance to be **safely** done on nodes within the system during full operation. Our Scale Computing HyperCore one-click, cluster-wide, rolling updates use automated virtual machine live migration across the cluster for planned node OS maintenance (and even reboots as required) without impacting production application availability or compromising data redundancy or failover capability **even temporarily**. You don't get that with two servers where doing planned maintenance intentionally takes down half of your paired system for a while.

The Edge Site Cost Multiplier and TCO

In a multi-site distributed environment, the impact of even a small cost increase or reduction is magnified across the number of sites. That makes it especially important to consider all costs from right-sizing initial hardware and software acquisition and deployment to risk and cost of application downtime and on-premises troubleshooting and "break-fix" support. Further, consideration should be given to future-proofing where possible to eliminate the next round of multi-site forklift/rip-and-replace infrastructure projects as hardware ages or business needs change unexpectedly or unexpectedly.

Many of the most relevant costs to consider have been mentioned already but to recap some of the major cost areas to consider:

- **Usable** compute and storage need available to run applications (excluding infrastructure overhead from RAM and CPU cycles for running the stack to RAID storage redundancy overhead)
- Cost of application downtime vs. cost of hardware and software solutions for downtime mitigation.
- Personnel cost in obtaining fluency in complex solutions
- Staff/contractor cost of on-site repair trips or on-site service contracts
- Day one - costs of deployment of the infrastructure hardware and software stack as well as applications (including migration of existing / legacy workloads if needed)
- Day two - costs of ongoing management / monitoring / updating across all locations

Scale Computing Platform architecture has been designed first and foremost to provide autonomous IT infrastructure that ensures your applications keep running with little intervention even as hardware components fail.

SC//Platform does this by efficiently pooling existing and new IT infrastructure resources (compute, memory, storage, network) at each location into a self-monitoring and self-healing system that manages application availability and system redundancy automatically.

By keeping your applications running and "self-healing" itself from ordinary failures, HyperCore significantly reduces the need to send costly IT support resources on site and can save both internal IT staff time and reduce the cost and reliance on third-party onsite service contractors.

SC//Platform was designed to operate the infrastructure with a fraction of the RAM/CPU footprint of other solutions allowing the use of significantly lower cost hardware than alternative approaches, leaving more of your hardware is available to applications vs. overhead. By distributing compute and storage across multiple smaller system, you can significantly lower your costs while enhancing overall reliability and performance of your applications. The savings can extend beyond just the cost of the hardware and warranty to lower power and cooling costs as well.

Lastly, SC//Platform was designed to simplify IT administration from initial deployment and startup to ongoing monitoring and operations and maintenance. Adding new applications or expanding resources is a breeze allowing your IT team to meet business needs more rapidly at lower cost.

In IT, "It Depends" - The Exceptions

Much of this paper has focused on the benefits of SC//Platform that begin with three or more nodes and the reasons why this configuration is the most popular and generally the most economical. But, it should be noted that SC//Platform can also be deployed in two-node configurations and even **single-node, fully managed systems with various levels of redundancy available**.

Scale Computing Platform can offer custom two-node cluster configurations with a very low-cost tiebreaker component that actually runs the same HyperCore software stack as standard nodes (and is therefore fully integrated with and managed with the rest of the cluster) - but optionally does not provide compute and/or storage resources to the system. One case where this might make sense is to make use of already deployed server pairs[5] - but update them with SC//HyperCore centralized management functionality and autonomous orchestration capabilities.

Further, Scale Computing even offers single-node edge systems with the full SC//HyperCore software stack that can be used to run applications independently at edge locations but with the ability to replicate to another cluster that could be at a remote/central data center or cloud, or even could be a second single node system in the same location, each replicating data to the other for redundancy.

These configurations provide the centralized management, monitoring, and deployment capability of SC//Platform and may be suitable for locations and applications that are not sensitive to downtime or that are "stateless" in that they do not create/update data that needs to be protected in real-time. Recovery in those cases might be to simply redeploy a new appliance and applications from templates. However, they do experience many of the same limitations with systems described above. For example, not only would a single SC//HyperCore node not allow immediate local failover in the event of a server failure, options like non-disruptive rolling upgrades with applications running are obviously not an option unlike clustered SC//HyperCore configurations.

Both of these configurations offer upgrade and expansion paths to higher level of functionality by adding additional SC//HyperCore nodes down the line to provide full three-node cluster functionality and benefits.

[5] SC//HyperCore is available pre-loaded on a wide range of existing Scale Computing appliance models as well as software only licensing for a wide range of existing servers. For large scale deployments Scale can test and certify many existing or preferred x86 based server / system configurations as part of a large-scale enterprise licensing arrangement.

Let Scale Computing work with you to create a custom TCO / ROI analysis for your environment, applications and your roadmap to see what configuration options make the most sense for you. Chances are good that you will be surprised, as in the vast majority of cases customers find that a properly sized and efficient, three-node Scale Computing HyperCore cluster meets all of their needs while providing better application availability and manageability for their edge locations at a significantly lower overall cost.

For more information on how to get started with Scale Computing Platform or if you have additional questions, contact Scale Computing at info@scalecomputing.com or call 877-722-5359.

Corporate Headquarters
525 S. Meridian Street - 3E
Indianapolis, IN 46225
P. +1 317-856-9959
scalecomputing.com

EMEA B.V.
Europalaan 28-D
5232BC Den Bosch
The Netherlands
+1 877-722-5359