Distributed Edge Computing:

Alternative Architectural Approaches

From the Experts at
Scale Computing, Inc.

SCALE COMPUTING

This paper explores architectural alternatives for distributed edge computing, from traditional ROBO (remote office/branch office) setups to modern hybrid and cloud-native applications. It compares the cost, performance, and operational trade-offs of enterprise-class servers, two-node standby clusters, and autonomous hyperconverged systems such as *SC//HyperCore™* virtualization suite, optimizing IT for edge computing network architecture.

Designing IT Infrastructure for the Distributed Edge

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 supporting current and future needs for new, modern application architectures (such as containers, cloud-native apps, IoT, and 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 Implications of Distributed Edge Deployments

Cost is a primary factor. In this multi-site, distributed edge computing infrastructure, even minor differences in CapEx and OpEx costs per location will multiply and massively increase the total cost of ownership. It's essential 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 likely have a single server (perhaps a glorified desktop machine) or two servers 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 accept the risk of inevitable downtime until properly trained help (and likely replacement parts) can arrive at the remote site.

Operational Challenges in Remote IT Management

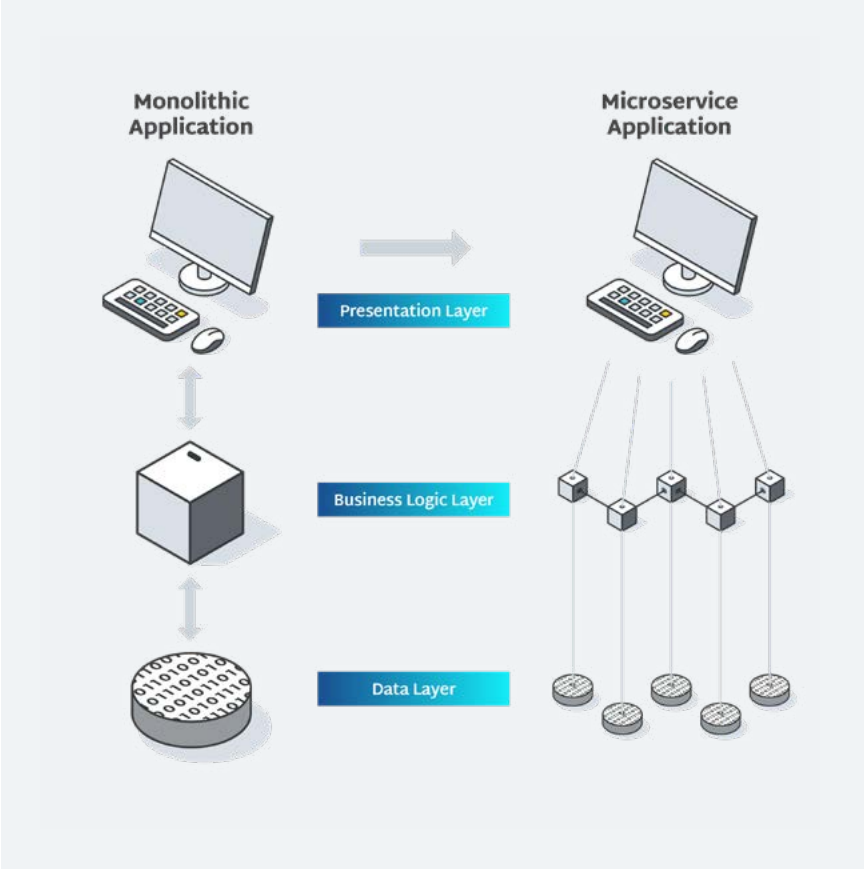
Many times, the remote office scenario is even less ideal, with centralized IT staff getting surprise calls (often outside working hours) from panicked application users or remote managers complaining that their application(s) are down and figuring out, on the fly, what to do: how to get someone there, 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 explore a range of infrastructure redundancy options available to remotely support the IT needs of a distributed edge computing architecture and present the associated costs, risks, and benefit trade-offs for each.[1]

Traditional Edge Infrastructure: Challenges & Limitations

The Limitations of Monolithic Edge Server Architecture

There is something to be said for the simplicity of older monolithic systems, where a single app runs on a single OS on a 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.



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.

[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 rollback in the event of undesirable changes such as ransomware, failed OS, or application updates or data corruption. SC//HyperCore data protection offers scheduled snapshot retention, remote snapshot replication to a centralized data center or cloud, as well as third-party backup software integrations to comprehensively address these needs across edge locations.

Server Fortification: Mitigating Hardware Failures at the Edge

The more critical remote applications are to the organization, considering the cost of downtime, lost productivity, and disruption, the more justifiable it becomes to invest in mitigating the most common and expected single points of failure (SPOFs) at each site. These investments are often necessary but come with added cost.

One of the most common SPOFs is disk storage. Disks not only fail frequently, but also often serve as performance bottlenecks. To address this, organizations typically implement:

RAID storage (Redundant Array of Independent Disks)

- Reduces the risk of a single disk failure bringing down applications
- Improves storage performance and availability
- Helps protect against data loss

Power failure is another frequent source of unexpected downtime. Hardware fortification typically includes:

Redundant power supplies and UPS systems

- Minimizes the impact of local power failures
- Keeps systems running during brief outages or until a safe shutdown is possible

Networking also presents a common SPOF, particularly when dependent on a single path or switch. To improve resilience and reduce downtime risk:

Dual network paths with redundant switches

- Improves network availability
- Supports load balancing and fault tolerance

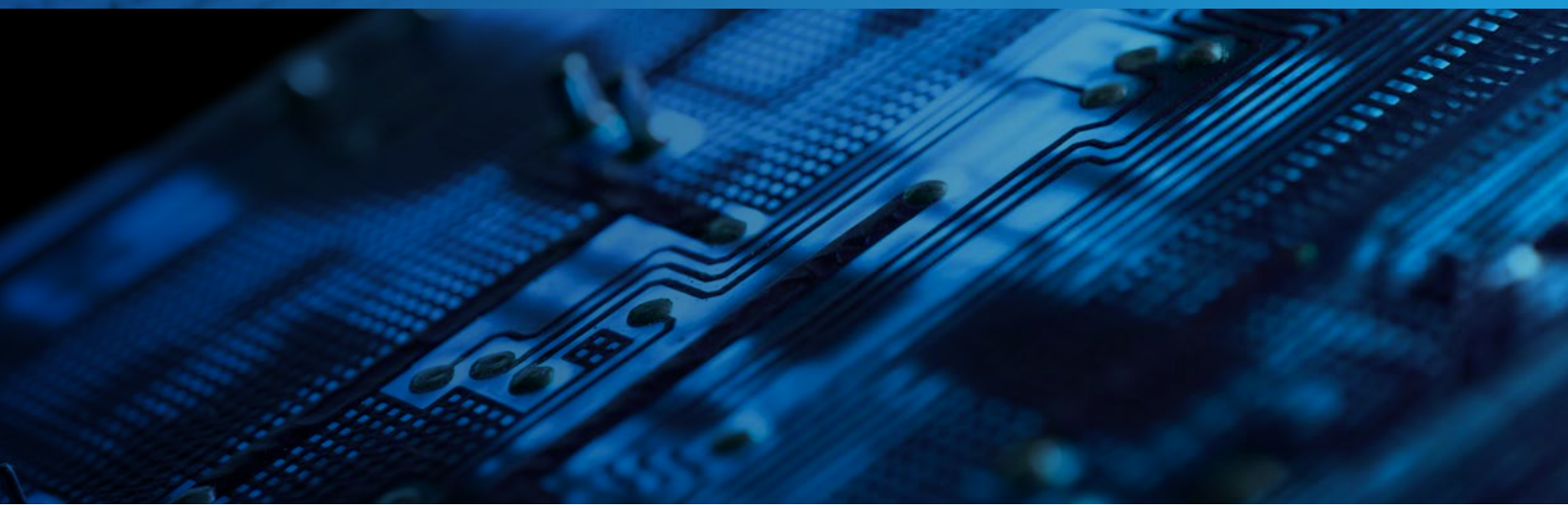
Despite all these reinforcements, monolithic server architectures still have inherent weaknesses. When a remote server experiences a failure, whether hardware or software, the entire stack can go offline. This triggers a troubleshooting process that typically requires someone with IT expertise, familiar with your specific environment, to either access the system remotely or travel onsite.

Diagnosis may involve testing multiple components (e.g., software, power, network, hardware), and in many cases, physical replacement of failed parts, such as CPUs, motherboards, RAM, or RAID controllers—many of which cannot be practically made redundant.

Crucially, even with robust local hardware redundancy, these efforts only protect the integrity of the single box. They do not ensure independent redundancy of your data. That's why periodic backup and restore processes remain absolutely essential. These are your safety net—required not if something fails, but when it does.

Redundant Server Pairs: High Availability at High Costs

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 idle or passive and that you hope you never need to use. This can result in a 100–120% increase in capital expenditures (CapEx), while operational costs (OpEx)—such as power, cooling, and maintenance—can rise by up to 50%, even for hardware that remains idle most of the time.

Many "bolt-on" systems or add-on features impose significant performance overhead on your applications, sometimes consuming enough resources to require 20–30% more compute capacity than originally planned, necessitating the purchase of even larger systems—twice the original capacity—to achieve the same level of work.

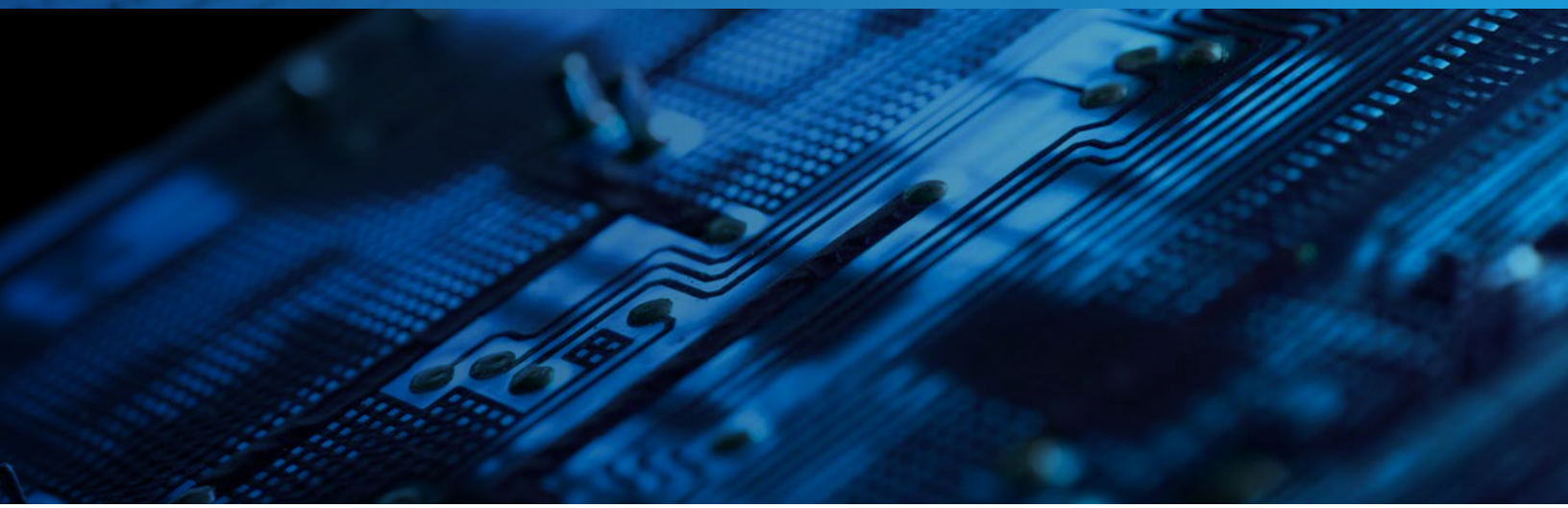
The Data Risk of Redundancy: Why Backup Alone Isn't Enough

Redundant server pairs can keep an application running, but after a failure, you are back to the state of the original single server, which is now fully responsible for continued operation and storage. When you lose access to your primary system and switch to your backup system, you also lose your backup plan and redundant storage location. Getting the primary system back online quickly remains a top priority and is 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 and remote mirroring of all data to the second system, resulting in "usable" storage capacities that are often less than 25% of the total raw capacity across both systems.[2]

[2] If each of two servers contains 4 x 2TB SSDs for a 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 of 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.



Feature	Primary Server + Backup	HyperConverged Systems
Initial Capital Expenditure (CapEx)	High: Requires purchasing duplicate hardware, including servers, storage, and networking equipment	Lower: Consolidates compute, storage, and networking into a single solution, reducing hardware needs
Operational Costs (OpEx)	High: Increased power, cooling, and maintenance costs due to separate systems	Lower: Simplified management and reduced infrastructure lead to decreased operational expenses
Redundancy	Limited: Redundancy is lost once the primary fails; backup becomes the single point of failure	Integrated: Built-in redundancy across nodes ensures continuous availability even during component failures
Storage Efficiency	Low: Dual redundancy methods can reduce usable storage to less than 25% of total capacity	High: Efficient data distribution and deduplication maximize usable storage capacity
Downtime Risk	High: Manual failover processes can lead to extended downtime	Low: Automated failover mechanisms minimize downtime during failures
Performance Overhead	High: Synchronization and failover processes can degrade performance	Low: Optimized resource allocation maintains consistent performance levels
Management Complexity	High: Managing separate systems increases administrative overhead	Low: Unified management interface simplifies administration
Scalability	Limited: Scaling requires significant investment in additional hardwares	High: Modular design allows easy and cost-effective scaling
Disaster Recovery	Complex: Recovery processes are manual and time-consuming	Simplified: Integrated disaster recovery features enable quick restoration

Deployment Considerations for Edge Infrastructure

Before exploring modern alternatives, it's important to understand what drives infrastructure decisions at the edge. Edge environments present a unique combination of constraints—technical, operational, and financial—that make infrastructure design especially critical.

Key deployment considerations include:

- **Capital Expenditures (CapEx):** Traditional mirrored systems and over-provisioned hardware drive up upfront investment, often with underutilized resources.
- **Operational Expenses (OpEx):** Distributed environments multiply ongoing costs for power, cooling, maintenance, and remote support.
- **Redundancy Requirements:** High availability is critical, but legacy systems rely on inefficient failover approaches that limit true fault tolerance.
- **Management Complexity:** Limited on-site IT staff and inconsistent configurations across locations increase the risk and cost of human error.
- **Scalability Constraints:** Adding capacity or expanding services at remote sites often requires significant re-architecture or full system replacement.

These challenges explain why legacy infrastructure often falls short—and why a more efficient, resilient approach is needed to support modern edge workloads.

A Better Way: Hyperconverged Solutions for Distributed Edge Computing

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

Distributed RAID Across Servers: Protecting Storage at the Edge

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) provide 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).

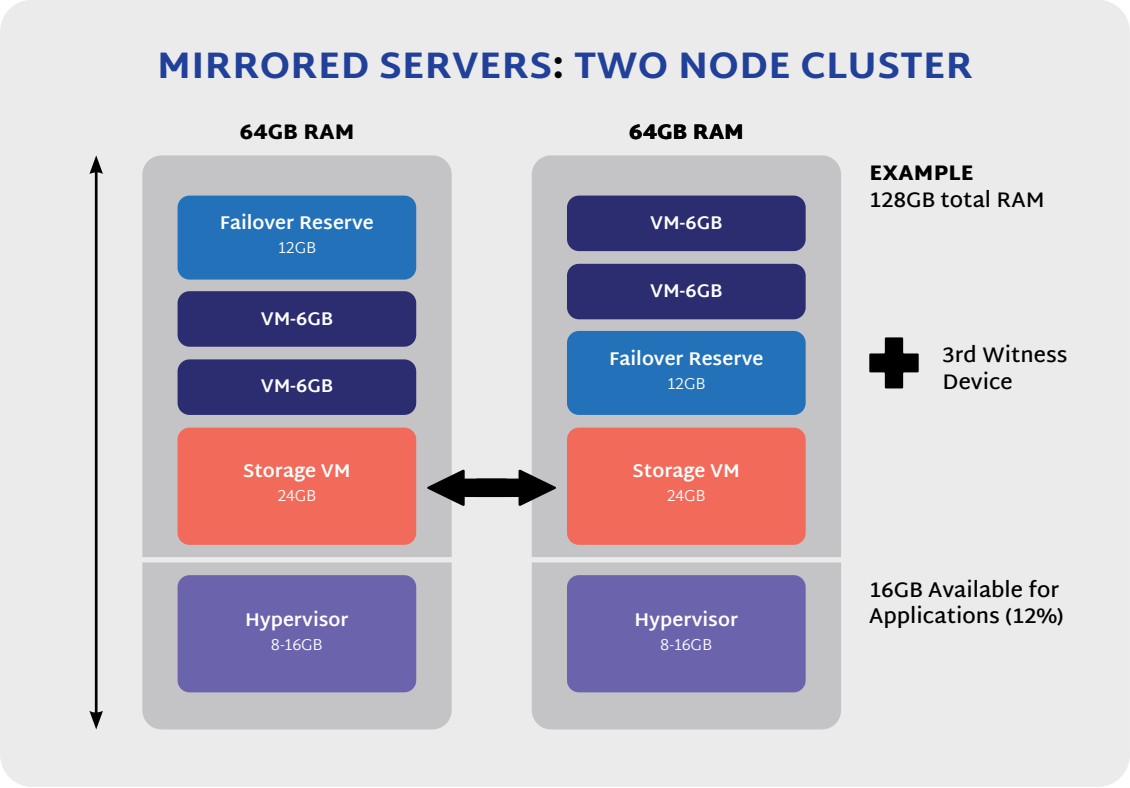
In the single monolithic server example above, it was mentioned that RAID is commonly required to protect data availability within a single server (as well as provide performance benefits vs. individual disks). What if the concept of RAID were extended to provide redundancy using disks located on different 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.

Server Cost Challenges: The Hidden Inefficiencies of Redundant Pairs

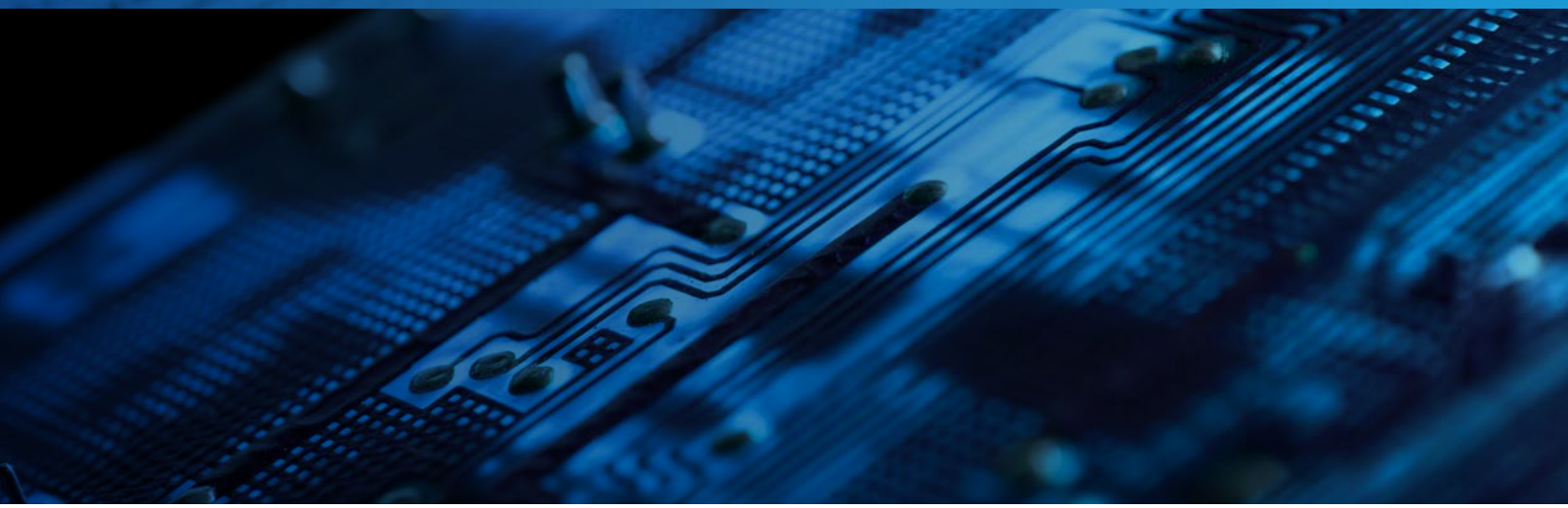
Some servers are far more expensive than others. If you've already priced out a powerful and resilient enterprise-class server (often paying more for a system with empty slots/sockets to allow for future scale-up expansion), buying a duplicate second server that just sits idle and ready to take over 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. Even in that case, half of the 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 mode.

Additionally, many infrastructure solutions that offer a "two-node cluster" or mirrored server pair are already inefficient in terms of resource usage. The operating system/hypervisor alone can consume a fair amount of resources. 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 a much bigger server than you anticipate because of the overhead needed to create the clustered redundancy.



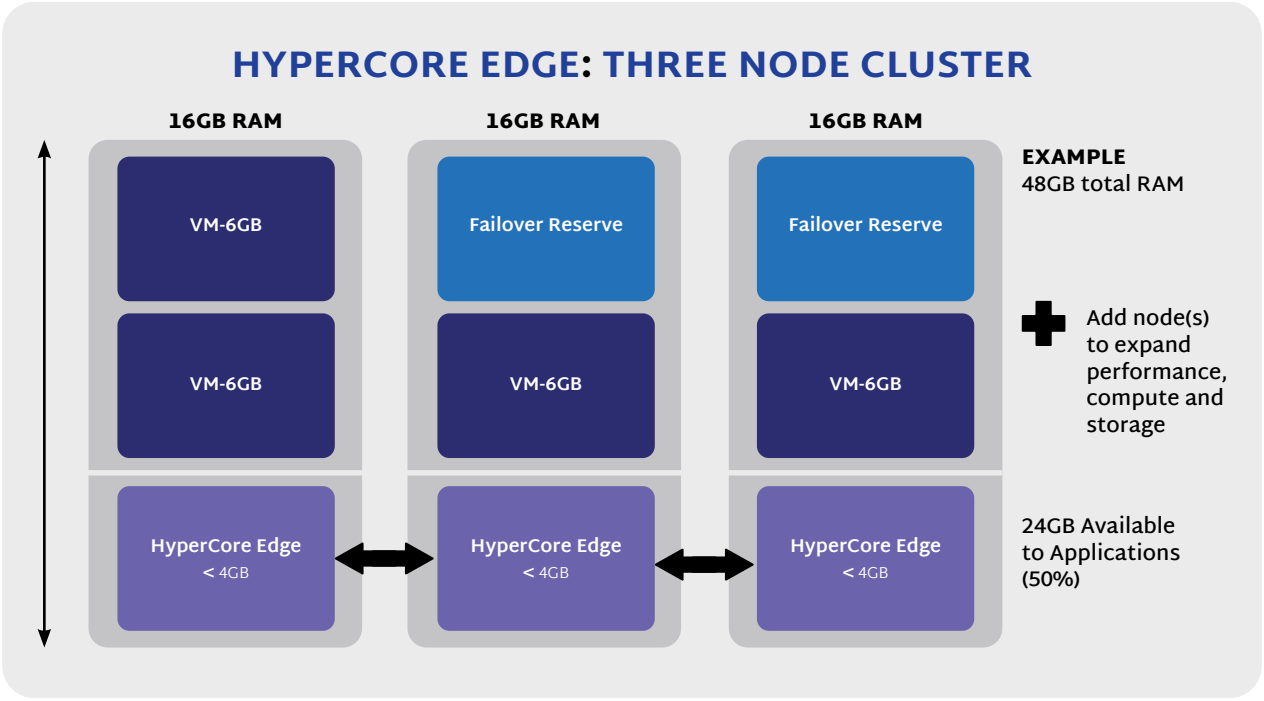
[3] In any so-called "2-node cluster" or mirrored pair server system, to allow a surviving server to positively prove that a secondary server is actually down and not still running applications or modifying data, some reliable system of preventing what is known as a "split-brain" cluster operation is required. This generally takes the form of some tiebreaker or witness device or devices used to verify which servers are actually up and which are actually down. If node 1 loses communication with node 2, it can contact the tiebreaker(s) to determine if the tiebreaker has visibility to node 2. Generally, each node is also 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. In some architectures, it actually becomes more critical than the compute nodes or servers themselves.



Three-Node Clusters: Higher Efficiency, Lower Cost for Edge Sites

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 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's worth of compute resources, not half the entire system. 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 efficient failover capability with three or more nodes, the system can continue to maintain full, distributed data redundancy for new or changed data even when a full node is 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 four-hour on-site repair service and locally stocked part depots goes away as well.

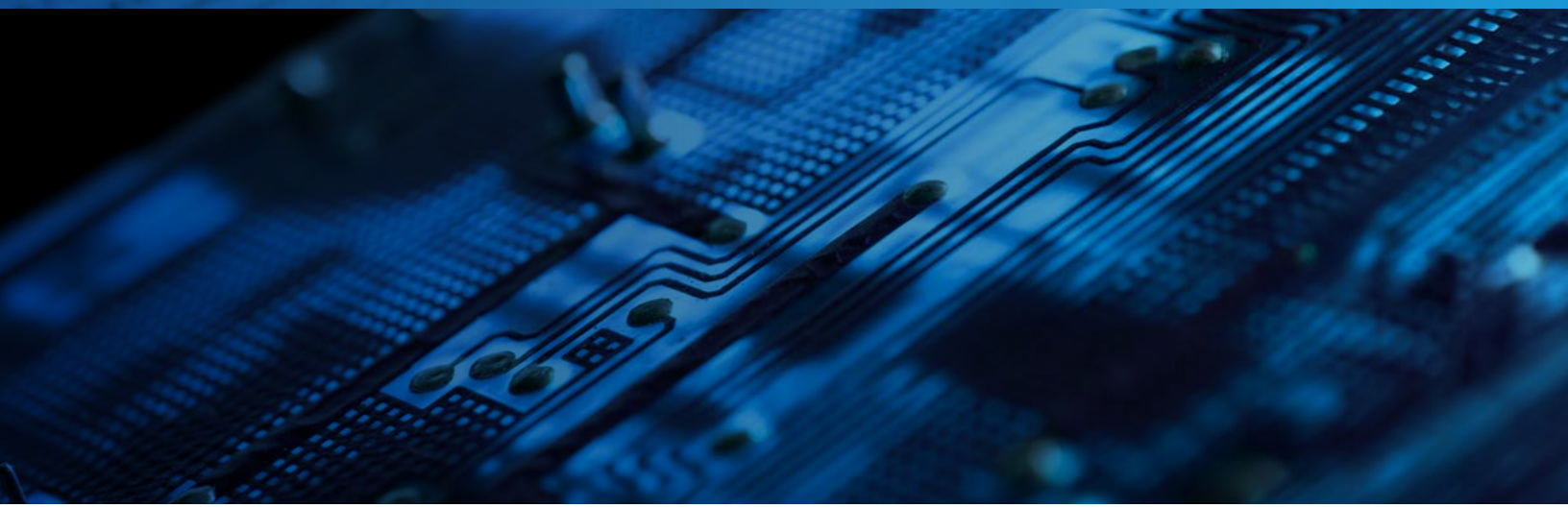
The inherent resiliency benefits of a three-node or greater distributed system allow a much wider range of cost-effective and efficient systems to be used. As a result, three nodes of commodity IT gear in a cluster can not only cost significantly less than a redundant pair of high-end servers, but they 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 to get just a little more compute capacity can often more than double the cost of the system and support.

The following table compares different redundancy models, highlighting why three-node hyperconverged clusters provide a superior balance of cost, resource utilization, and fault tolerance for distributed edge environments.

Redundancy Model	Server Count	Resource Utilization	Cost Efficiency	Failure Tolerance
Single Monolithic Server + RAID	**1**	Low: Only one compute node; RAID protects only the storage	Moderate: Lower CapEx but high downtime risk	Low: Single point of failure for compute/network; storage protected via RAID only
Two-Node Mirrored Pair (Active/Passive)	**2**	Low: One node remains passive/idle until failure	Low: High CapEx per usable resource; 50% hardware underutilized	Moderate: Tolerates one failure, but recovery can be slow and resource-wasteful
Three-Node Hyperconverged Cluster	**3**	High: All nodes active; shared workloads	High: Balanced CapEx with better utilization and protection	High: Tolerates one node failure with no downtime; efficient failover and data safety
Scale-Out Hyperconverged Cluster	**4+**	Very High: Linear scalability of compute and storage	Variable: Higher CapEx but strong ROI at scale	Very High: Multiple node failures tolerated depending on cluster size and config

[3] 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 constraints 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 CPUs. Running those same 4 VMs on a single server would require a 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: Seamless Growth for Distributed Edge Locations

With scale-out hyperconverged systems like SC//HyperCore clusters, which allow additional nodes to be added seamlessly, expanding pooled compute and storage resources as needs grow, the need to purchase expensive servers with unused or underutilized internal expansion capabilities 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), but 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 redeployed 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//HyperCore 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 SC//HyperCore software 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.

Reducing Total Cost of Ownership (TCO) in Distributed Edge Computing with Hyperconverged Systems

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 the risk and cost of application downtime, 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.

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

- Usable compute and storage need to be 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

SC//HyperCore architecture has been designed first and foremost to provide an autonomous IT infrastructure that ensures your applications keep running with little intervention, even as hardware components fail.

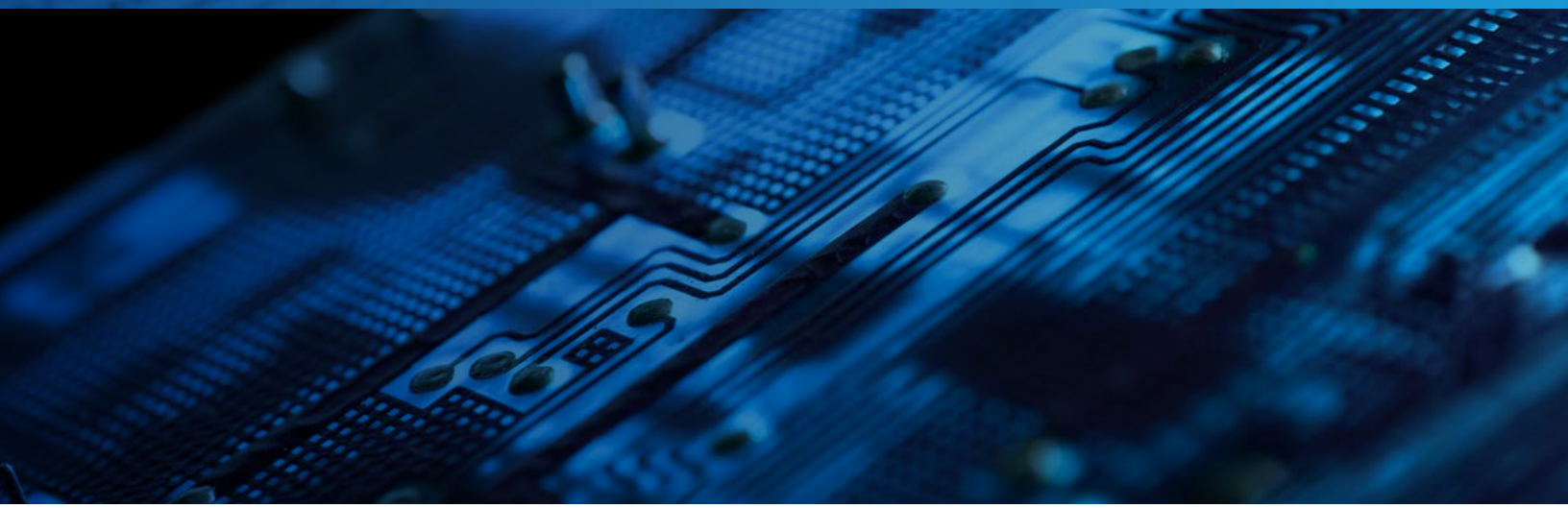
SC//HyperCore clusters achieve this by efficiently pooling existing and new IT infrastructure resources (compute, memory, storage, and network) at each location into a self-monitoring and self-healing system that automatically manages application availability and system redundancy.

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

It 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 available for applications versus overhead. By distributing compute and storage across multiple smaller systems, you can significantly lower your costs while enhancing the 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, it was designed to simplify IT administration from initial deployment and startup to ongoing monitoring, operations, and maintenance. Adding new applications or expanding resources is a breeze, allowing your IT team to meet business needs more rapidly at a lower cost.

To better understand the cost advantages of hyperconverged systems, the table below contrasts typical cost factors between two-node mirrored servers and three-node hyperconverged clusters.



Cost Breakdown - Two-Node vs. Three-Node Edge Infrastructure

Cost Factor	Two-Node Mirrored Servers	Three-Node HCI
Hardware CapEx	High: Requires duplicate servers with idle/passive hardware; often includes additional external storage and failover systems	Moderate: Shared resources across nodes with no passive hardware; converged compute and storage lowers overall system footprint
Power and Cooling	High: Passive node still consumes power and cooling resources, often with underutilized efficiency	Moderate: All nodes actively contribute to workloads, improving energy efficiency per unit of compute/storage
Maintenance and Support	Higher: Separate systems and infrastructure result in more complex maintenance and service agreements	Lower: Unified platform simplifies support and typically includes centralized vendor support
Resource Utilization	Low: Only 50% of compute resources used in normal operation; storage often mirrored, reducing usable capacity to <50%	High: All nodes are active; data is distributed efficiently with improved redundancy and usable capacity
Dowtime Migration Costs	High: Failover often manual or semi-automatic, with risk of longer downtime; application migration during failure is more disruptive	Low: Built-in self-healing and automated failover reduce disruption and downtime-related losses
Scalability	Limited: Scaling requires full duplication (adding 2 more nodes at a time), with significant cost and architectural complexity	Easy: Nodes can be added incrementally; software-defined scaling simplifies planning and cost control
TCO Over Multi-site Deployment (10+ sites)	Very High: Cost multiplies rapidly due to redundant hardware and underutilized infrastructure across sites	Lower: Efficient use of hardware, simplified management, and centralized orchestration reduce total cost across many locations

Flexible Deployment Options: Single-Node and Two-Node Edge Configurations

Much of this paper has focused on the benefits of SC//HyperCore virtualization that begins 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 it can also be deployed in two-node configurations and even single-node, fully managed systems with various levels of redundancy available.

SC//HyperCore™ virtualization suite also supports a 2-node cluster with a physical witness, giving organizations that have already ruled out a full three-node cluster the same high-availability behavior without the cost, power, or footprint of a third compute node. Rather than dedicating a full server to quorum, a lightweight witness device participates only in cluster quorum; it runs no storage and hosts no virtual machines, so it requires only modest hardware: as little as 4GB of RAM, an x86 processor, and enough storage to install the operating system. If one of the two compute nodes goes down, the cluster automatically restores quorum and resumes operation in roughly two minutes, with no manual intervention required and no dependency on a cloud-based witness.

One case where this might make sense is to use already deployed server pairs[5], but update them with SC//HyperCore centralized management functionality and autonomous orchestration capabilities.

Further, Scale Computing 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 be a second single-node system in the same location, each replicating data to the other for redundancy.

These configurations provide centralized management, monitoring, and SC//HyperCore deployment capabilities 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 may involve simply redeploying a new appliance and applications from templates. However, they do experience many of the same limitations with the 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, but 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 a higher level of functionality by adding additional SC//HyperCore nodes in the future, providing full three-node cluster functionality and its associated benefits.

[5] The SC//HyperCore OS 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 Computing can test and certify many existing or preferred x86-based server/system configurations as part of a large-scale enterprise licensing arrangement.

Why Hyperconverged is the Future of Distributed Edge

Modern edge infrastructure must be more than just functional—it must be resilient, cost-effective, and scalable. Hyperconverged systems, like SC//HyperCore virtualization, are designed specifically to meet these needs across diverse edge environments.

- Hyperconverged systems reduce downtime risks by eliminating single points of failure and enabling automated failover and recovery.
- Three-node clusters offer an ideal balance of performance, cost efficiency, and fault tolerance for edge deployments.
- SC//HyperCore platform delivers scalable, autonomous edge infrastructure, minimizing management overhead across thousands of locations.
- Flexible deployment models support everything from micro-branches to larger edge data centers, adapting to a wide range of IT requirements.

Contact Scale Computing for a custom TCO analysis and see how hyperconverged edge solutions can simplify operations, improve uptime, and reduce costs at scale.

CORPORATE HEADQUARTERS

3307 Northland Dr #500 // Austin, TX 78731
P. +1 317-856-9959 // **scalecomputing.com**