Technical White Paper

Don’t Believe your HCIs

Three Potential Risks of an HCI Architecture

Don’t Believe Your HCIs

Three Potential Risks of an HCI Architecture

The appeal of HCI as a concept is that by bringing compute, network and storage together in a fully-tested, controlled environment, infrastructure administrators would be freed from the challenges of integrating point solutions and would have scalable, guaranteed performance at lower risk. For specific workloads such as VDI and ROBO, the theory is that customers can allocate resources quickly, scale easily, and reduce costs significantly because of the integration and features in its platform. The reality is more complicated—it’s difficult for HCI to deliver scalability, simplicity, and cost advantages without sacrificing performance. In the “3 reasons to beware the hype around hyperconverged” paper, we outlined a broad array of hidden risks with HCI. In this paper, we will double click on the next level of details with concrete data and dive deep into some specific workload comparisons.

Tintri has taken a different approach than HCI. Our architecture was built around modules to allow for seamless scaling and flexible integration with external infrastructure. Tintri storage delivers the industry’s broadest support for individual VMs while providing leading analytics and tight integration with a variety of hypervisors. Unlike HCI, by optimizing storage and supporting tight, simple integration across multiple compute and networking platforms, Tintri gives users industry-leading performance, powerful scalability, and flexibility to dynamically optimize across storage, compute, and networking.

Let’s look at 3 key complications that HCI can introduce into an organization’s infrastructure: Cost, risk, and change. Some of the difference among architectural choices can be subtle at first glance but yield significantly different results.

Cost

A key appeal of HCI is lower cost. By integrating and optimizing storage, compute, and networking, HCI is supposed to reduce deployment and management costs. But it’s not that simple.

Support and Integration

Support for these integrated systems often comes at a premium, due to higher complexity of troubleshooting (see below). And although it might seem odd to say integration is a cost with a “fully integrated” HCI, unless it is deployed in a greenfield site it can be exceedingly difficult and expensive to deploy it into an existing infrastructure. So while you get a single throat to choke if there is a problem, you often pay for the privilege.

Overprovisioning

HCI's modular structure enhances scalability in some cases while complicating it in others. Best practice is to align CPU, memory, and storage configuration to avoid storage hotspots. But, for example, for very data-intensive applications, storage resources are in higher demand than compute. Since HCI vendors recommend balanced (homogeneous) nodes, to add more storage you also have to add more compute, driving up costs unnecessarily. This is known as the HCI tax.

Licensing

Because HCI resources are built around balanced, integrated nodes, dedicated services to support local storage, and their associated licensing costs will scale with storage as more nodes are added. In addition, Microsoft SQL Server and Oracle database licensing is based on the number of CPUs per node. As described above, this means that overprovisioning, often needed for higher performance, drives higher socket-based and core-based licensing costs. Since databases require dedicated storage on each node, more nodes are often needed for enough vCPUs for all database instances.

Vendor lock-in

It may seem odd that vendor lock-in would be an issue for an HCI platform that leverages commodity hardware but once deployed, it is basically a rip and replace if you want to switch to a different vendor. Software and infrastructure is tightly coupled, hindering you from mixing/matching platforms from different vendors to optimize across your infrastructure.

The combination of more expensive support, architectural constraints driving the HCI tax, bloated licensing costs, integration costs, and vendor lock-in can easily offset any perceived pricing advantage of HCI. Customers have found that HCI doesn't always save them time or money, which was one of their purchase justifications. For example, Shire Pharmaceuticals switched from HCI to Tintri and Cisco UCS, resulting in a 75% reduction in operating expenses.

Cost Comparison

	Total Servers	CPU Cores	Memory	Storage	Total Cost
Tintri + Cisco	36	1,008	18,432 GB	495 TB	$1,470,860.00
HCI	48	1,024	24,576 GB	1,305 TB	$5,213,853.00
	Overall more compute resources			**2.6x more raw storage**	**3.54x more expensive**

Tintri + Cisco = Predictable Performance with less resources at 75% less cost than HCI

Tintri's approach to infrastructure avoids HCI pitfalls by enabling users to optimize their resource mix across servers and virtualization-centric storage. Support costs are typically lower than HCI, you purchase only the capacity you actually need, licensing costs and storage requirements are minimized, deployment is simple, and you can pick (and switch) best of breed compute and storage resources. This lets you more effectively ride the innovation curve for each technology and minimize obsolescence. By providing flexibility in provisioning, Tintri users both lower their costs and enhance their ability to optimize and manage their infrastructure.

Risk

One concern buyers have with HCI is unpredictable performance, as this can drive increasing risks. HCI systems don't always have the latest components, which can prevent mission-critical applications from operating at peak performance. This is especially true when comparing to shared all-flash-array (AFA) storage systems where IOPS, predictability and failure domain are often problems for HCI.

IOPS

To maximize performance, standard AFAs use 28-40 cores/controller for 13-24 SSDs. Going higher can impact performance predictability by impacting IO density. However, HCI apportions available cores across servers and storage, resulting in much of the power of AFA being unavailable. For example, vendors have claimed high IOPS based on a burst of reads coming from cache. If you burst writes that exceed their buffer, or need IO not in cache, or run into contention, you must pull data from slower storage media, so performance suffers and requires you to add more nodes. This is because of the limitation on CPUs available for storage. Typical limits are 8 vCPUs or 20% of available CPU. This impedes the performance of 6-24 flash drives on each node, wasting flash IOPS. To make things worse, data reduction takes away even more vCPUs, which is why it is often optional. If you do end up adding more CPUs to support storage performance, you incur higher licensing costs (see above), so you end up paying application licensing fees for storage CPUs.

Predictability

Unless you are limiting HCI to low-tier, highly-time-out-tolerant applications, performance predictability is often as important as latency and IOPS. But with HCI, storage and compute run on the same node, with storage software running as a virtual appliance on a virtual machine at the guest layer. Because CPU resources are shared, even moderate utilization can cause a major bottleneck with HCI. If you try reserving CPUs, per-VM reservations can have significant system effects, including cluster-wide HA failover policies. Even reservations don't guarantee instant CPU access for a virtual appliance. You can still have IO delays if there is another vCPU scheduled that is finishing up its operations.

Failure domain

With HCI, resources are tightly coupled. This creates domino effects around maintenance and node failures. For maintenance, HCI:

- Creates potential new bottlenecks and noisy neighbor problems from evacuating data before maintenance and spreading all node loads across other nodes
- Constrains clusters due to reduced storage from node maintenance
- Reduces available flash, especially as cache in a hybrid system. This increases flash misses and reduces performance.

So HCI forces users to choose between enduring the risks of performing maintenance vs the risks of not performing maintenance. In addition to maintenance risk, HCI systems also expose users to increased failure risk. Host failure:

- Reduces resources for compute and storage
- Reduces flash for hybrid systems. Failed node flash must be rewarmed and existing VM data gets evicted from the cache.
- Causes the entire process to be repeated when the failed node is reintroduced to the cluster. A single component failure can cause an entire node to collapse. HCI's tight resource integration creates significantly greater operational risk compared to decoupled systems.

All of these factors impede predictability and can have a major impact on application performance. Busy nodes or clusters can cause unexpected spikes in latency. Wide latency variation across IOs can cause applications to time out or fail. Multiply this effect in an enterprise cloud environment with thousands of VMs/containers and HCI "web scale" claims are called into question. Worse yet, if you want to leverage useful services like snapshots, deduplication, or compression, you have to either add more hardware or face worse predictability and performance.

If performance is a priority, AFAs offer a better alternative by delivering better latency of each IO operation, boosting total IOPS, and offering better IO performance predictability. External storage is better at delivering each of these compared to HCI.

Taking this a step further, Tintri's architecture automatically assigns each VM or container its own lane to avoid resource conflict. It simplifies setting min/max QoS for individual VMs and guarantees app performance. Compare this with HCI, where getting high performance requires an expensive compromise.

A note on data locality

HCI vendors claim that data locality minimizes network traffic and storage IO latency by keeping data in local flash vs pulling it from over a network. While the traffic benefit is real, if the latency of pulling data from another node is not a significant contributor to overall storage latency, the benefits of data locality can be negligible. For example, Cisco Nexus switches have latencies under 1 µs. This is negligible relative to the latency generated by the HCI system itself when requests for data must be translated into the data's location within the HCI distributed file system. If a hypervisor is involved, that adds more latency. A final point is that while modern flash has very high IOPS, concentrating high I/O storage on a few local flash drives can lead to contention and noisy neighbor problems.

Change

One of the biggest reasons that HCI has gained attention are claims that those systems significantly reduce infrastructure complexity. In reality, the situation is more nuanced. If you have a static infrastructure, this may be true. But that's almost never the case. When you introduce changes like data movement, daily changes, and scale, you also introduce significant challenges with HCI.

Latency

HCI is designed to store multiple copies of each block of data. This significantly boosts IO latency, especially when mirroring data across a network to other nodes. Erasure coding imposes a further latency penalty (post process erasure coding may only be available for cold data). In ESG lab tests, even the best HCI had 5ms latency, significantly worse than AFAs. A large benefit of distinct storage and compute resources is that VMs are easily moved across hosts using vMotion or live migration. Latency is further degraded by data movement and HA events, which reduce available resources and introduce overhead.

Troubleshooting

While integrating compute and memory in one node eliminates the need to manage and optimize the interface between the two (reducing complexity), it also makes troubleshooting more challenging. The tight coupling of memory and compute on each node hinders fine-grained isolation to identify any bottlenecks.

Ok, so to address an identified symptom, you increase a VM's memory and CPU resources. What if that doesn't solve the problem? It's probably IO.

- Is the bottleneck host, network, or storage?
- Are too many services increasing metadata to degrade performance?
- Is the problem a guest VM or storage VM? Shared resources complicate things.
- Is the problem IO to internal or external storage nodes? If internal, does throttling or migration of workloads fix or worsen performance? If external, is it a network or node issue? Are there multiple external nodes containing VM data?

Managing this troubleshooting challenge is difficult with a small HCI deployment and only gets worse at scale.

Maintenance

HCI has automated much of the maintenance process—it happens in the background. But the performance impact may be very much front and center. As a node is taken down for maintenance, all data is automatically evacuated onto other nodes. If those nodes do not have sufficient headroom to absorb this additional data, application performance may suffer, or worse it could cause node failure. As noted above, administrators are forced to choose between the risks of maintenance vs the risks of no maintenance.

Upgrades

Like with maintenance, installing upgrades on HCI involves taking nodes offline, pushing data to other nodes, again degrading system performance and potentially impacting resource availability. With HCI, you are forced to decide between having the latest capabilities and potentially reducing system availability and performance during the upgrade cycle.

Workload comparison

To better understand the potential drawbacks of HCI, it might be helpful to look at some specific examples. VDI was one of the earliest use cases for HCI and continues to be an area where they see success. But if you look deeper, you will see that HCI brings with it significant capacity and licensing "taxes" relative to Tintri that should be included in any cost calculation.

Side by Side 1000 VDI VMs Capex Analysis

	Total Servers Needed	Total Desktops/ Server	Total Physical Cores	Total Memory (GB)	Total Raw Storage (TB)	Actual Storage Bought
Tintri	7	160	280	5,376	34	34
HCI	10	112	400	7,680	120	230
	HCI 1.43x more	**HCI 1.43x more**	**HCI 1.43x more**	**HCI 1.43x more**	**HCI 3.52x more**	**HCI 6.76x more**

HCI Compute Requirements: 1.44x more than Tintri
HCI Storage Requirements: 6.76x more than Tintri
HCI Socket Licensing Cost: 1.44x more than Tintri
HCI Consistent sub-ms performance: No

Upon closer examination, HCI imposes a number of hidden costs. For the 2 cases above, relative to Tintri, it requires nearly 1.5x more compute, and 3.5x storage, nearly 1.5x socket license costs, all without delivering consistent sub-ms performance.

Conclusion

The above discussion highlights significant considerations that should be examined when considering HCI. Obviously, HCI is not a broadly flawed solution. For some workloads, particularly those that aren't latency sensitive, are smaller scale, and are relatively easy to troubleshoot, HCI may be worth considering. In addition, they make it easy to add nodes, although it is notably difficult to remove them.

The problem is that fundamental tradeoffs in the architecture can have dramatic, negative consequences across an infrastructure. For all its touted virtues, HCI exposes customers to added cost, risk, and challenge around change. By imposing constraints meant to simplify infrastructure design and maintenance, HCI creates other problems that can outweigh its benefits.

Tintri's approach is to provide the performance, predictability, flexibility, and cost structure to deliver many of the same benefits of HCI while avoiding many of the drawbacks. With Tintri, you get both flexibility and simplicity, performance and predictability, and the latest technology with bullet-proof integration. Our best-of-both-worlds platform avoids HCI's costly and dangerous tradeoffs without adding frustrating complexity.