Evaluating Data Locality

Is keeping data local really key to HCI performance?

A DeepStorage Technology Report
About DeepStorage

DeepStorage, LLC. is dedicated to revealing the deeper truth about storage, networking and related data center technologies to help information technology professionals deliver superior services to their users and still get home at a reasonable hour.

DeepStorage Reports are based on our hands-on testing and over 30 years of experience making technology work in the real world.

Our philosophy of real world testing means we configure systems as we expect most customers will use them thereby avoiding “Lab Queen” configurations designed to maximize benchmark performance.

This report was sponsored by our client. However, DeepStorage always retains final editorial control over our publications.

Copyright © 2017 DeepStorage, LLC. All rights reserved worldwide.
Evaluating Data Locality

Introduction

As the flash revolution took hold in the storage industry, we at DeepStorage observed that the systems that took the fullest advantage of flash all used metadata drive data layouts, which also optimized data services such as snapshots and clones. Comparing those systems to conventional storage systems, we observed that data layout is destiny. Today, the developers of some hyperconverged systems are promoting data locality, another variant in data layout, as a key advantage.

In this paper we’ll look at the advantages and disadvantages of data locality in some detail to allow customers to decide if data locality is right for them.

What Do We Mean By Data Locality?

Some vendors of hyperconverged infrastructure have designed their systems to preferentially store a full copy of the data for each virtual machine on the local storage of the host where that VM runs. This is data locality. Keeping the storage for each VM on the same host as the compute makes sense on an intuitive level; reading data from a local device avoids a network hop and storage architects avoid any I/O we can. Enforcing data locality also satisfies the urge users have to know where their data lives physically.

Some HCI solutions implement strict locality, which always maintains a local copy of the VM’s virtual disks on the host with the VM—even if that restricts migration of the VM exclusively to those hosts currently holding replicas of that VM’s data for resiliency.

An HCI Solution with data locality

Other systems use a more loosely constrained version of data locality that allows migration to any host by moving data from the VM’s former host to the new one only as blocks in the virtual disks are accessed.

Data Locality’s Promoted Advantages

The vendors that promote data locality claim that it is the only way to deliver the performance that modern data centers demand. Let’s look at the logic they use to justify these claims and see if data locality really is a requirement for high performance, or just one path to the promised land.
Evaluating Data Locality

The biggest of these claimed advantages is that data locality minimizes storage I/O latency. Because a full copy of each volume is stored on a local SSD, all read I/Os can be directed to that local media. This eliminates the latency of sending a request across the network to another node and waiting for a response. Writes will still require I/O across the network to protect data against a node failure, but a significant fraction of write I/Os will also go to the local media.

Accessing remote storage media will indubitably have more latency than accessing that same SSD or disk drive in the local system. The question is whether the latency added by the network hop is a significant portion of the system’s total storage latency.

**Network latency is lower than you think**

To hear the advocates of data locality talk, network technology hasn’t advanced significantly since the days of 1Gbps Ethernet and switches with latencies of a millisecond or more. In reality, today’s standard top of rack Ethernet switches have latencies of under two microseconds, while specialized ultra-low latency switches like Cisco’s Nexus 3548 have latencies under 1μs.

<table>
<thead>
<tr>
<th>Device</th>
<th>Port Bandwidth</th>
<th>Channel Bandwidth (MBps)</th>
<th>Device Bandwidth (MBps)</th>
<th>Latency (read)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nexus 5596UP</td>
<td>10Gbps</td>
<td>1250</td>
<td></td>
<td>1.8μs</td>
</tr>
<tr>
<td>Nexus 3548</td>
<td>10Gbps</td>
<td>1250</td>
<td></td>
<td>350ns</td>
</tr>
<tr>
<td>Nexus 92160</td>
<td>25Gbps</td>
<td>3125</td>
<td></td>
<td>700ns</td>
</tr>
<tr>
<td>End to end IP connection 10Gbps Ethernet</td>
<td>10Gbps</td>
<td>1250</td>
<td></td>
<td>8-20μs</td>
</tr>
<tr>
<td>Samsung PM863 SATA SSD</td>
<td>6Gbps</td>
<td>750</td>
<td>520</td>
<td>320μs</td>
</tr>
<tr>
<td>NVMe SSD</td>
<td>3.94GB/s (3.0x4)</td>
<td>3950</td>
<td>3000</td>
<td>200μs</td>
</tr>
<tr>
<td>SAS SSD</td>
<td>12Gbps</td>
<td>1500</td>
<td>1400</td>
<td>300μs</td>
</tr>
<tr>
<td>3D Xpoint Optane SSD</td>
<td>7.88GBps</td>
<td>3550</td>
<td>2400</td>
<td>10μs</td>
</tr>
</tbody>
</table>

*Table 1 – Bandwidth and Latency of Common Devices*

As you can see in Table 1 above, the latency of even an NVMe SSD at about 200μs is ten times the 20μs or so it takes for a message to travel from one node to another across a 10Gbps Ethernet link, including the TCP/IP stacks at both ends. Because the HCI solution will use the same software to locate the data it needs and to interface with the media on each node, accessing data on a remote node should have 20-50μs more latency than accessing local media.

It’s also important to bear in mind that the HCI solution’s storage software also adds a significant amount of latency. When an application makes a request to read some data, the software stack has to translate that request for some block or byte range within a virtual volume to the actual location of the data within the HCI solution’s distributed file system or object store. Only once this translation is complete can the HCI solution actually access the storage media.
Evaluating Data Locality

If the HCI solution relies on the hypervisor for access to the media, accessing the media will also require the host to perform one or more context switches between the storage VM and the hypervisor, which adds additional latency. Context switch latency can be minimized by giving total control of the storage controller(s) to the storage VM via VMdirect path, but “file system” latency is unavoidable.

While we at DeepStorage like a good architectural argument as much as anyone, just as the proof of the pudding is in the eating, the proof of storage performance is in the testing. While we aren’t aware of any public benchmark results for a system with data locality, Storage Review tested VMware’s vSAN 6.2 and saw average latency of approximately 900μs while performing 361,000 4K read IOPS on a four-node cluster.

Reduced network traffic

One indisputable advantage of data locality is reduced network traffic. If the storage I/O performed by the virtual machines on a host that averages 70% reads can be satisfied from the local drive, it will certainly generate significantly less network I/O than if it accessed data spread evenly across the system’s many nodes.

When considering this additional network traffic it’s important to note that a system that doesn’t place data locally will still place some data on the local storage. In a four-node cluster, a system creating two copies of its data will place one copy of the data on the local drive one fourth of the time. This fraction decreases as the cluster size increases providing more potential targets.

Our friend and sometime associate Alistair Cooke wrote a series of blog posts where he calculated the amount of network traffic for systems with and without data locality when storing two and three replicated copies of each VM’s data. His math shows that in clusters of 8-16 nodes, a system with data locality could generate as little as one third as much network traffic as a system without locality.

While it’s easy to call a reduction in network traffic a virtue, it will only really matter if the network connecting the various nodes in a hyperconverged system becomes the system’s bottleneck.

Our version of Goldratt’s theory of constraints tells us that we should only be concerned about how much our system is consuming its most constrained resources. In fact, we frequently trade greater use of an abundant resource to preserve the dearest resource in the system. Think how a storage system trades CPU cycles for storage space when performing data reduction.

The important question isn’t whether data locality reduces network traffic, but whether the network in an HCI system is sufficiently constrained that we should care.

Today’s HCI systems typically use two 10Gbps Ethernet connections for each node, with 25Gbps just on the threshold of going mainstream. If, for the sake of argument, we limit internode storage-related traffic to just one of the two connections, we’ll have 10Gbps of full duplex bandwidth at each node. This leaves another 10Gbps of full bandwidth for non-storage traffic, from user access to the servers to vMotion.
Evaluating Data Locality

To fully saturate that 10Gbps link would require the storage system to perform more than 200,000 4K IOPS per node at 70% reads on a system storing three copies of each piece of data. The full 16-node cluster would have to deliver over 3 million IOPS to flood the net.

Our conversations with HCI users also supports our conclusion that network bandwidth is generally not a constrained resource; most HCI clusters run out of CPU and/or memory long before the network becomes a bottleneck. With 25Gbps Etherne now coming to market at only a 15% premium to 10Gbps we can’t see network bandwidth being the bottleneck for the foreseeable future.

The downside to data locality

Like any other feature an architect may add to the design of a storage system, data locality comes with some baggage. Much of that baggage comes in the form of complexity in the distributed file system, or object store, that the HCI solution builds from the node’s local storage.

One supposed advantage of HCI is that it simplifies storage management by replacing 20 separate pools of storage from 20 LUNs to a single datastore. If HCI systems enforced strict locality, administrators would have to perform the same jigsaw puzzle machinations to place VMs on hosts with enough storage that they’re running from with 20 LUNs. If an administrator needed to create a VM with more storage than is available on a single node, they’d be out of luck.

Most HCI vendors avoid this nightmare by allowing a VM’s data to “spill over” to remote storage exclusively if there isn’t enough local space for one copy. While that lets a system administrator manage the system as a single pool of data, optimizing performance still requires either the system administrator, or the HCI solution’s automatic placement system, to factor storage space remaining on each node into their decisions.

I/O concentration

The core argument for data locality is that reads from the VMs on a host are satisfied by the storage, most frequently an SSD or two, on the node that provides compute for that VM. While today’s SSDs can deliver a significant number of IOPS, satisfying I/Os from local storage concentrates that I/O on the local SSD. Thus, several high-storage I/O virtual machines running on the same host can create contention for the local storage, along with the resulting noisy neighbor interference.

By distributing data across their nodes, rather than hoarding in locally, an HCI solution reduces the creation of I/O hotspots and the inconsistent latency the resulting contention can create. Just as storage arrays use wide striping solutions like VMware’s VSAN spread the read I/O load across multiple devices preventing a bottle neck at the local SSD when multiple VMs on the same host hit their storage hard at the same time.

Constraints on VM placement for VM movement/load balancing

As we’ve already seen, enforcing data locality complicates VM placement because administrators must account for available space and performance on the host’s storage. Things get even more complicated as we start moving virtual machines from host to host with vMotion and its automated big brother DRS (or heaven forfend, a node should fail).
Evaluating Data Locality

One leading HCI system enforces locality by limiting vMotion to either of the two servers that holds a replica of that VM's virtual disks. While the VM replicas for any host are spread across all the VMs in the cluster, this constraint ultimately leads to lower VM density to “leave room” for vMotion.

---

Data Locality Limits VM migration

Most others will hide the underlying data locations from vMotion and DRS, allowing the VM to be migrated to any host in the cluster. When the VM starts running on a new host, data is migrated from the original host, and/or any surviving replicas, as it’s read by the VM. Some systems will migrate the rest of the VM’s data as a background task, while others will leave any cold data on the original node.

On a system with loose data locality that only copies data as it’s accessed, a single VM’s data will be spread across several hosts, removing the warm feeling of knowing exactly where their data is stored from those that like that sort of thing.

It’s important to note that any effort to re-establish data locality, by necessity, increases the total amount of I/O the system performs. Even if the system doesn’t immediately start copying all of a VM’s data as soon as it’s relocated, re-establishing locality adds a write to local storage to every read I/O it performs to fetch data from a remote node.

The problem for data locality is that the corporate data center is becoming ever more cloud-like and dynamic. A few years ago, administrators would have a few hundred VMs that all ran 24x7. Sure, they might vMotion machines once every few months to perform maintenance on their hosts, but for the most part VMs stayed put and admins load balanced by Excel spreadsheet.

But static VMs require enough excess capacity on each node to accommodate peak load. More efficient operators have moved to dynamic load balancing through solutions like VMware DRS. In the emerging development model that uses micro-services, containers are spawned and destroyed by an orchestration layer such as Docker Swarm or Kubernetes as needed, and may only run a few seconds for each instantiation.
Evaluating Data Locality

Today’s orchestration processes, including Kubernetes and VMware DRS, don’t know anything about data locality. They see the HCI data pool as one resource and can’t, for example, migrate a VM to a host that already has a copy of that VM’s data. When you have containers popping up on random hosts, running for a minute and going away, data locality is going to be a random thing, just like on systems that don’t consider locality at all.

Data locality and modern data storage

Data locality is pretty easy to understand when we’re talking about an old fashioned storage system where every .VHD or .VMDK file was an independent entity replicated to a few remote nodes for protection. You remember the kind where creating five clones of a 15GB Windows template took an hour and 75GB of disk space because it actually made five copies.

None of us wants to go back to those bad old days. We like that our modern storage uses a metadata clone, not a physical copy of a template. That way it only has to store the common data once.

Data Locality and Metadata Clones

Now imagine an HCI system that supports data locality and metadata cloning, as many do. On that system we have a production MySQL server. Our developers have complained about issues not showing up with the small section of the production database; since the metadata clones don’t take up much space we’re going to create eight clones of that VM so each of them can work on their own full copy of the production data. We’ll protect those clones with just one replica--after all, if the data’s lost we can just create new clones.

When our developers go to work, these eight MySQL VMs will be running on five hosts. This, of course, creates a conundrum. If these MySQL VMs are really metadata clones, there will be only two copies of the common data and the VMs running on those two hosts will run significantly faster than the VMs running on the three hosts that lack a local copy.

An HCI solution could solve this apparent performance problem by making additional copies of the data on the three hosts. In fact, at least one HCI solution will do this in special cases like the root images of VDI linked clones, where the common data image is read-only.

While creating additional copies can address I/O spikes such as VDI boot-storms, those additional copies consume storage capacity, which reduces the system’s overall efficiency.

Data deduplication

Data deduplication, like the clones we discussed above, uses metadata to abstract one copy of a piece of data into all the places that data appears. If we set a data protection policy of three copies, the HCI system should create three, and only three, copies of each unique data block.
Evaluating Data Locality

If the data being stored is reducible, like the Windows update patches in the user’s delta disks in a persistent VDI environment, several VMs will point to the same set of data. That data will be local on three nodes and remote to VMs on any other host.

There’s an inherent conflict between data deduplication, which reduces the total amount of data that’s stored, and data locality’s need for a local copy. As with the metadata clones, a system could store additional local copies but that would sacrifice much of the efficiency gained by data deduplication.

Erasure coding

Our discussions so far have assumed that HCI systems protect their data by replicating the .VMDK and/or .VHD files that make up a virtual machine across a total of two or three nodes. Replication is, however, quite inefficient with 50% overhead for 2-way replicated data and 67% for 3-way replicated data.

HCI solutions have recently added erasure coding to improve their storage efficiency. These systems stripe data with parity across nodes in the HCI cluster much like the way RAID systems stripe data across disk drives. Several HCI systems use four data strips and two parity strips to provide a similar level of resiliency to 3-way replication. Replacing 3-way replication with 4D+2P erasure coding doubles the amount of data the system can store.

Once again we find a conflict between data efficiency and data locality. The whole idea of erasure coding is to restrict the amount of data on each node to limit the impact of a device failure. Locality concentrates each VM to the host it runs on, which means the system has to store more data elsewhere to protect itself.

Data locality and offline resources

Virtualization, or more specifically vMotion, vastly reduced the number of sleepless nights and endless weekends suffered by the storage administrators of the world. Where we used to have to carefully schedule a downtime window for 4AM on St. Swithun’s Day, now we can vMotion workloads to other hosts and update firmware the same day important security patches are issued.

When a host goes offline temporarily for maintenance, or more permanently because of a failure, an HCI solution has to accommodate the compute workloads from the offline host while simultaneously rebuilding storage resilience.

If data locality really is central to the performance of an HCI system, then relocated, or restarted, VMs will run slower until the system has re-established locality for its hottest data. The extent to which this may be noticeable by the users, if at all, remains untested.

HCI systems that leverage locality distribute data across nodes at the virtual disk level. Their rebuild is a 1 to 1 process, or with triple mirroring a 2 to 1 process, that stresses storage nodes with existing replicas. The new host is therefore additionally burdened
Evaluating Data Locality

running the new compute workload and writing a new replica of its data. If the original host was only offline for a few minutes, that data will be written yet again as the VM migrates once more.

Relieved of any need to maintain locality, HCI architects can make other decisions that reduce the impact of systems going offline. They can provide an option to postpone the beginning of their rebuild process to accommodate hardware maintenance, a feature we recommend only in conjunction with higher underlying resiliency such as 3-way replication. They can also stripe data in smaller stripes across more nodes, turning the rebuild into a more efficient many-to-many operation with a smaller impact on performance.

Conclusions

At first glance, data locality seems a common sense approach to optimizing an HCI system’s storage performance. However, the closer we look at how data locality actually works, the less attractive it looks.

Data locality may provide a performance advantage for some systems under normal conditions - third-party benchmarking of VMware’s vSAN shows it can deliver average latency of 800μs without the discussed tradeoffs of data locality. This pushes any advantage locality can provide into the realm of diminishing returns.

The problem is that any perturbation of the system, even one as simple as vMotioning a VM, breaks locality. These systems will, therefore, run in a degraded mode more frequently, and the performance impact of that degraded mode will be more severe than on systems that don’t depend on locality.

Any performance advantage that locality may provide comes with some significant costs in both storage efficiency and the consistency of that same performance. VMs migrated away from their data, or to servers with other storage-heavy VMs, will run slower as the local SSDs bottleneck on the additional traffic. Enforcing locality prevents these systems from load balancing I/O intensive virtual disks across hosts.

Data efficiency features such as data deduplication and erasure coding work their magic by spreading data across multiple nodes in the HCI system, which is antithetical to maintaining data locality. The greater efficiency these technologies provide can reduce the cost of an all-flash system with deduplication and erasure coding to match the price of a hybrid system with locality. We believe that users would be better served by the consistent performance of an all-flash system rather than gaming the performance of a hybrid system with data locality.

While advocates of data locality may tout the technique as a way to achieve the performance that today’s data centers demand, in many cases this performance comes at the expense of other data center advancements including VM mobility, deduplication, erasure coding, and emerging development architectures built around containers and microservices.