Book of Basics - Hyperconverged Platform

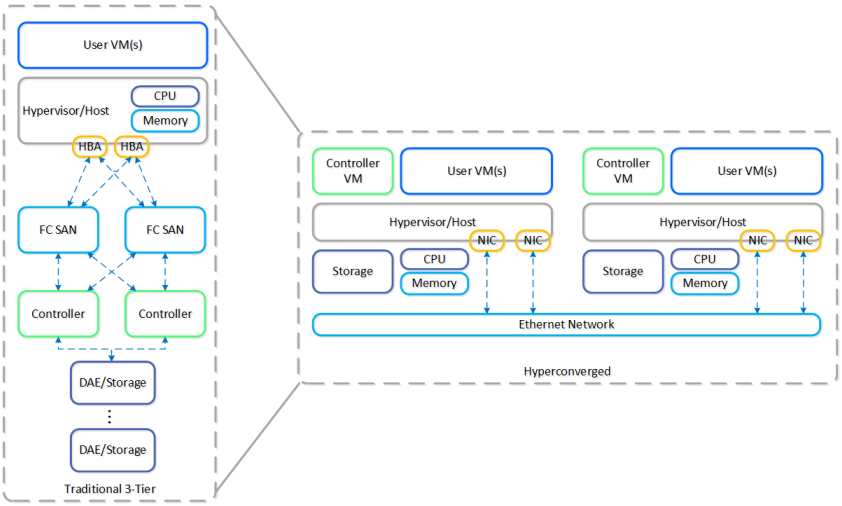
[PDF generated December 08 2021. For all recent updates please see the Nutanix Bible releases notes located at https://nutanixbible.com/release_notes.html. Disclaimer: Downloaded PDFs may not always contain the latest information.]

For a video explanation you can watch the following video: LINK

There are a few core structs for hyperconverged systems:

- Must converge and collapse the computing stack (e.g. compute + storage)
- Must shard (distribute) data and services across nodes in the system
- Must appear and provide the same capabilities as centralized storage (e.g. HA, live-migration, etc.)
- Must keep data as close to the execution (compute) as possible (Importance of Latency)
- Should be hypervisor agnostic
- Should be hardware agnostic

The following figure shows an example of a typical 3-tier stack vs. hyperconverged:



3-Tier vs. HCI

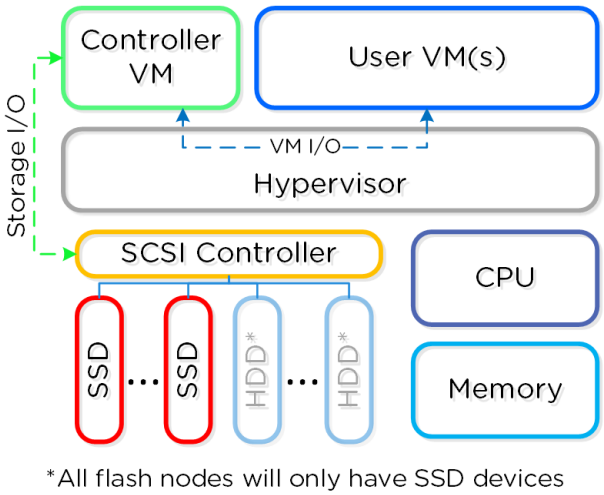
As you can see, the hyperconverged system does the following:

- Virtualizes and moves the controllers to the host
- Provides core services and logic through software
- Distributes (shards) data across all nodes in the system
- Moves the storage local to the compute

The Nutanix solution is a converged storage + compute solution which leverages local components and creates a distributed platform for running workloads.

Each node runs an industry-standard hypervisor (ESXi, AHV, and Hyper-V) and the Nutanix Controller VM (CVM). The Nutanix CVM is what runs the Nutanix software and serves all of the I/O operations for the hypervisor and all VMs running on that host.

The following figure provides an example of what a typical node logically looks like:



Converged Platform

The Nutanix CVM is responsible for the core Nutanix platform logic and handles services like:

- Storage I/O & transforms (Deduplication, Compression, EC)
- UI / API
- Upgrades
- DR / Replication
- Etc.

NOTE: Some services / features will spawn additional helper VMs or use the Microservices Platform (MSP). For example, Nutanix Files will deploy additional VMs, whereas Nutanix Objects will deploy VMs for MSP and leverage those.

For the Nutanix units running VMware vSphere, the SCSI controller, which manages the SSD and HDD devices, is directly passed to the CVM leveraging VM-Direct Path (Intel VT-d). In the case of Hyper-V, the storage devices are passed through to the CVM.

Virtualizing the Controller

The key reasons for running the Nutanix controllers as VMs in user-space really come down to four core areas:

1. Mobility
2. Resiliency
3. Maintenance / Upgrades
4. Performance, yes really

Since the beginning we knew we were more than a single platform company. In that sense, choice has always been a big thing for us, whether it is with hardware, cloud or hypervisor vendors.

By running as a VM in user-space it decouples the Nutanix software from the underlying hypervisor and hardware platforms. This enabled us to rapidly add support for other hypervisor while keeping the core code base the same across all operating environments (on-premise & cloud). Additionally, it gave us flexibility to not be bound to vendor specific release cycles

Due to the nature of running as a VM in user-space, we can elegantly handle things like upgrades or CVM "failures" as they are outside of the hypervisor. For example, if there is some catastrophic issue where a CVM goes down, the whole node still continues to operate with storage I/Os and services coming from other CVMs in the cluster. During a AOS (Nutanix Core Software) upgrade, we can reboot the CVM without any impact to the workloads running on that host.

But isn't being in the kernel is so much faster? Simple answer, NO.

A common discussion topic is the debate around being in the kernel vs. in user-space. As a matter of background, I recommend reading the 'User vs. Kernel Space' section which covers what both actually are and the pros and cons of each.

To summarize, there are two areas of execution in an operating system (OS): the kernel (privileged core of the OS where drivers may sit) and user space (where applications/processes sit). Traditionally moving between user-space and the kernel (aka context switch) can be expensive in terms of CPU and time (~1,000ns / context switch).

The debate is that being in the kernel is always better / faster. Which is false. No matter what there will always be context switches in the guest VM's OS.