Docker: What do Storage Pros need to know?

by George Crump, Lead Analyst

Docker was created to solve the problems that organizations face when they implement server virtualization on a wide scale; overhead and inefficiency. These challenges occur because virtualization is a sledgehammer to the problem it was designed to solve; allow multiple applications to run simultaneously on the same physical hardware in such a way that if one application fails the rest of the applications are not impacted. This is the real goal of virtualization, isolation of applications so that a misbehaving application does not impact another application or its resources.

The other "goals" like consolidation, mobility, improved data protection are not goals at all, they are just outcomes of achieving the real goal. The problem is that virtualization places two significant taxes on the data center in achieving its primary goal of isolated applications. These taxes have led to a new form of virtualization, called containers, and Docker is the container solution getting most of the attention.

The Virtualization Taxes

The first tax that virtualization places on the data center is that it requires an entire server to be virtualized in order for it to achieve the isolation goal. The problem is that what most data centers actually want is application or operating system isolation, not server isolation. In order to achieve application isolation, the entire operating system and the server's hardware are virtualized. The impact of hundreds of virtual machines, each running a duplicate set of base operating system processes, consumes memory and CPU resources.

The second tax is the performance overhead of the hypervisor. Applications simply will not perform as well when virtualized as they would on bare metal systems. For a great many applications this overhead is a non-issue because they don't require any more performance than what the hypervisor can give them. For some, however, the performance impact can be noticeable, especially when the application needs to interface with something external to the hypervisor like networks or storage. In these instances the hypervisor needs to translate between the abstracted virtual world and the real data center. The use of more powerful processors can offset this hypervisor overhead but it is still less efficient than the native, bare metal use case.

The Docker Container

Docker and other container technology is essentially application virtualization. It creates a virtual instance of the application or even parts of that application and isolates it by creating multiple copies of the operating system's users space, the location in which applications normally run. But all the containers run on the same server. Docker allows each container to

share the operating system memory and processes so they do not need to be run separately for each application. Today Docker is Linux based and runs Linux applications, but support for Windows is on the way and Microsoft is engaged with the Docker community. In the meantime, companies like DH2i are creating container technologies for Windows applications. There is also work going on an open container project that promises compatibility between various container implementations. No matter what environment is common in the data center, containers will be an option and storage administrators need to be prepared for their arrival.

Docker and Storage

A Docker container is designed to be temporal, something that spins up, executes some tasks and then disappears. Its overlay file-system implements a copy-on-write methodology. Essentially there is a core operating system and then containers “tree” off of that. This non-persistent nature has lead most Docker implementations to leverage local storage and a very basic feature set. But as use of the technology grows, organizations are looking to have containers that can communicate with each other and access each other’s data. As data within a Docker environment becomes more permanent, customers are looking for more robust data storage services.

Fortunately, Docker provides a -v directive that allows the mounting of remote shared volumes. This capability seems to be expanding as evidenced in some of the recent experimental releases. Bootable images can be stored on these volumes. But again, other than its overlay file-system, Docker is relatively simplistic in its storage capabilities. This means that containers have to be made responsible for file access and locking control as they exchange data.

What to Look For in Shared Storage for Docker

Hyper-scale Storage for Hyper-scale Docker

Docker was created to run distributed applications, which means the underlying infrastructure should be designed to run in a distributed architecture. As a result Docker is almost exclusively used in hyper-scale environments. Even as the enterprise adopts Docker they will likely also use it for their internal cloud / hyper-scale needs. Ideally the storage infrastructure should match the design of the compute infrastructure by delivering a hyper-scale storage solution suitable for these environments. This means that the storage systems should be able to incrementally scale performance and capacity by adding nodes to an existing storage cluster.

Many of these scale-out architectures run Linux at their core. In the future, they may be able to take advantage of containerization and run Docker containers directly on the storage system. This may be useful for times when direct access to data is needed.

Docker is often highly automated using Restful APIs, and the storage infrastructure should be extensible via similar APIs. Most often, this automation is used to script provisioning and decommissioning functions within the Docker infrastructure. It would make sense then for the storage system to be a part of this automation process. Essentially, as a container is provisioned the appropriate storage resources are provisioned at the same time and within the same script. When the container is decommissioned the storage system should allow storage resources to be released back to the global pool as part of the decommissioning script.

Data Services

A Docker storage solution should also fill in the gaps in data services. As mentioned above Docker storage capabilities are very basic. As Docker is deployed more frequently, especially

as it makes its way into the enterprise, IT professionals will require a similar set of storage services comparable to what they are used to from other environments. It is important that the storage systems provide these data services capabilities like deduplication, replication, snapshots and clones, without compromising cost effectiveness or the system's ability to scale.

Scale Performance and Capacity

A storage platform that supports Docker will also need the ability to scale both in capacity and performance. This is necessary because Docker use cases run the gamut from containers that process large big data sets to containers that are more transactional in nature. The system needs to have the ability to tier data between performance storage and capacity storage. It also needs the ability to continuously add nodes without disruption. A key factor in performance delivery will be the efficiency of the cluster software itself, and how it manages the numerous nodes. When a Docker environment reaches its compute scale the storage node count may grow very high, and how well the storage software manages that cluster will be critical not only to performance but also total capacity.

Multiple Access Methods

To support Docker a storage system should have the ability to be accessed by multiple protocols. This is especially important given that Docker is still in its infancy and how it is accessed may change. Right now the storage system should provide native block, file and object access. Other environments may want to program directly against the storage backend using REST or RPC, skipping more traditional protocol access all together. Finally the storage systems should also allow a "Dockerized" access, where a container is created specifically to provide other containers with access to storage.

A Software Foundation

Finally, it is important that the Docker storage system has a software foundation, even if the vendor chooses to deliver hardware and software as a bundle. The advantage of software focus is that it gives the vendor flexibility to adapt to the Docker environment as it moves from infancy to maturity. How Docker handles storage will change dramatically over the next few years and the storage system needs to adapt to those changes as quickly as possible.

Without this flexibility a storage system may get stuck in a Docker 1.0 limitation when Docker 3.0 rolls out. This is similar to how storage systems were impacted as VMware evolved from almost no storage services to a complete set of storage services.

Conclusion

Docker is the shiny new toy for IT to play with but, like VMware before it, the chances are high that container technologies like it will move quickly from test labs, to test/dev, to full scale production. IT professionals need to begin investigating storage systems that will meet the early stage needs of the Docker environment while at the same time evolving to meet the longer term production class demands.