Excelero

NVMesh®

THE SECRET SAUCE IN NVMesh

PAPER

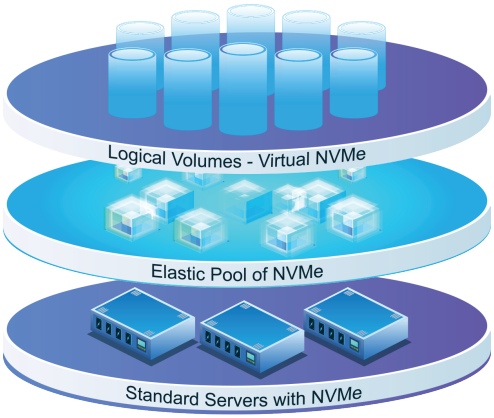
INTRODUCTION

NVMesh is a software-only product: you bring the hardware and we bring the performance. Excelero gives you full flexibility to select the hardware of your choice: standard off-the-shelf NICs, standard servers and NVMe flash devices. By choosing your own hardware, you can design the environment that works best for you to meet your objectives: the most cost savings, the best efficiency, the highest performance...

Performance used to be hardware-driven and many software-only storage solutions come with a performance trade-off. This is not the case with Excelero NVMesh, on the contrary: we allow customers to bring their own hardware and we give them better performance (than proprietary All-Flash Arrays) and lower overhead (CPU & storage utilization). NVMesh provides more IOPs, higher bandwidth and lower latency than any other solution on the market. Essentially, we only add five microseconds of latency to what you get from a local NVMe flash device.

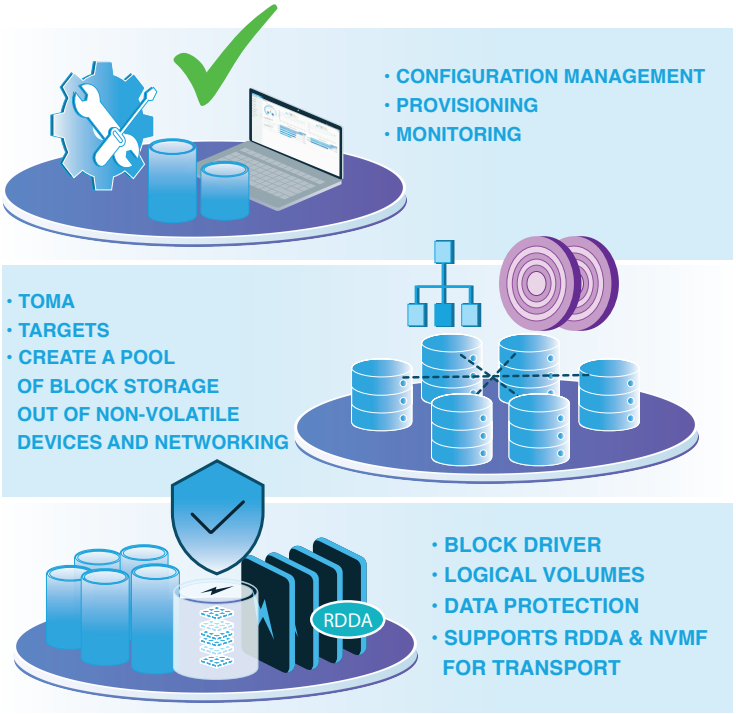
Before we get into the Secret Sauce, let's do a quick recap of what NVMesh is:

Just like server virtualization technology was created to maximize utilization of compute resources, NVMesh virtualizes NVMe flash devices to get maximum utilization of those devices. We abstract the capacity of your NVMe devices across your datacenter and create a pool of high-performance storage. From that pool, NVMesh creates virtual NVMe devices that can be spun up and utilized by any host at arbitrary size and protection level. Those virtual NVMe devices look and perform like a physical local NVMe device, with very little latency. Your applications can use the virtual devices as raw block volumes or with a high-performance file system on top of it (local or parallel).



100% SOFTWARE-ONLY

The NVMesh architecture is divided into three software planes: the Management Plane, Control Plane and Data Plane. We also have four essential software components, which implement these planes. The Management software maps nicely to the Management Plane. The Control Plane features the Target software and Toma, the Topology Manager. The Data Plane is entirely handled by our (Intelligent) Client Block Driver. The actual implementation works as follows: take two servers, one with an NVMe device, the other is an application server and they are connected over a high-speed network - let's say it's a 100 Gb Ethernet. Toma, our topology manager, and the target software run on the server with the NVMe device. On the application server we run our client software, The Client Block Driver.



The connection between the client (application) and the target (NVMe device) is made through our RDDA protocol (Remote Direct Drive Access). RDDA is similar to NVMe-over-fabric but with lower latency and no target side CPU overhead. NVMesh is also a complete storage solution, whereas NVMe-over-fabric is just the protocol and nothing more. **Fans of NVMe-over-fabric can use NVMesh with NVMe-over-fabric instead of RDDA, but keep in mind that RDDA is the real secret to the low latency and CPU overhead in NVMesh.** It leverages high-performance networking (100Gb) and ROCE V2 (RDMA Over Converged Ethernet). So in essence, RDDA uses the RDMA protocol on top of a standard Ethernet fabric.

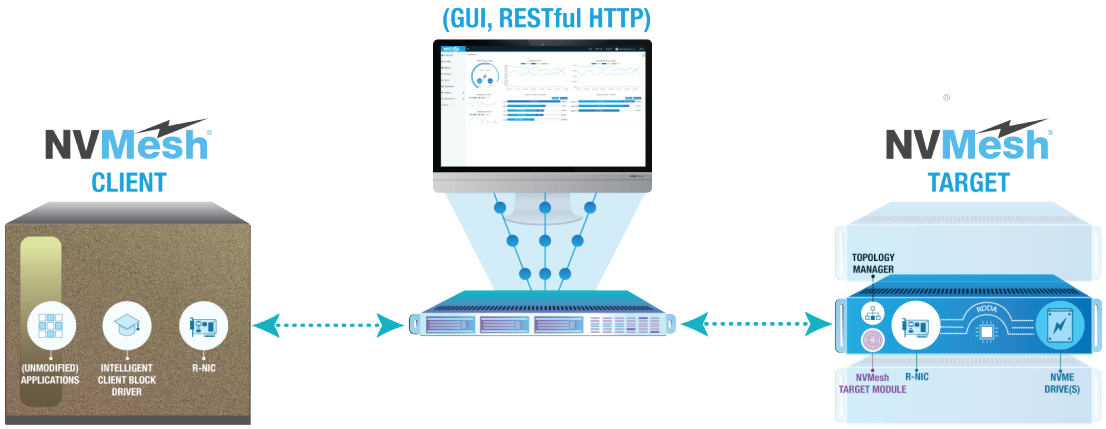
Back to our servers in the example: when the Client Block Driver establishes a connection, the Management acts as an authoritative database holding all definitions of the entire cluster and all logical volumes. In this case, we have a very simple logical volume, which uses a portion of our NVMe device. This definition is stored in the database. So once you define a logical volume, the Client Block Driver asks Management for the definition of this device. This logic runs in the Management Plane, there is no other communication with the Control or Data Planes. The Client has all the information it needs and now we will progress to the Control Plane.

The Client Block Driver now has the definition it needs and can now talk to Toma and the Target as part of the Control Plane. Assuming that everybody agrees on the definition, the Block Driver now establishes a connection from the Block Driver directly to the NVMe drive.

Once established, we move to the Data Plane, where all activities are handled by the Client Block Driver. You will notice that NVMesh is missing a central metadata controller or lock manager. These controllers are known to create performance and scalability bottlenecks and NVMesh was designed to not have any bottlenecks: NVMesh is 100% distributed. The intelligence lies in the clients, which understand exactly how a logical volume is laid out and how it's utilized. The client also has full control over the Data Plane (which blocks move where). Thus as you increase clients, you increase your ability to perform more IO.

LOW LATENCY & LOW OVERHEAD

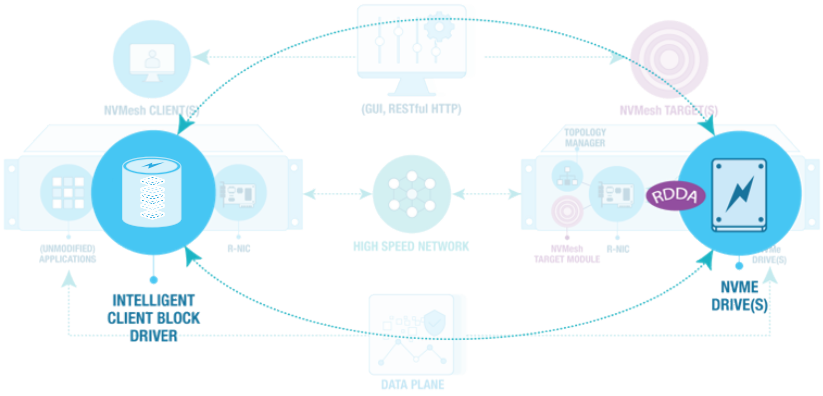
Once the client has the established connection with a target, it can directly send IO to and from this device. To do that, it leverages the RDDA Protocol. Let's take an example of 4KB writes: the client sends the data, the data lands in the host's memory (via and RDMA transfer) and at that point our NIC triggers an operation that tells the NVMe device to read data directly from memory and place it on the device. Without RDDA, the NVMe device would send a notice to the CPU when this is completed, which would consume CPU cycles. Thanks to RDDA we can reroute the completion notice to the NIC, which will send the response all the way back to the client over the network in one round-trip operation. **This is the magic of RDDA, which enables extremely low latency.** An entire round-trip operation adds about five microseconds of overhead over the response time is of an NVMe device: if a given device would write at 20 microseconds locally, an application writing through an NVMesh volume remotely would require 25 microsecond response time to that same device.



NVMesh does not use any CPU cycles on the target host because the client talks directly to the target host memory through the NIC, and triggers operations directly to the NVMe device and back. This is what allows NVMesh to have a revolutionary first - zero CPU overhead on target systems. Practically, this means that the server that hosts NVMe devices can serve out millions of IOPs to clients but does not use any CPU on the system. The CPU can be used for applications, file services or converged. In a disaggregated setup, with dedicated storage servers, NVMesh allows you to use storage servers with very inexpensive CPU's. Either way you are getting the lowest latency in the industry and the lowest overhead: low network overhead, latency overhead, and CPU overhead.

NO NOISY NEIGHBORS

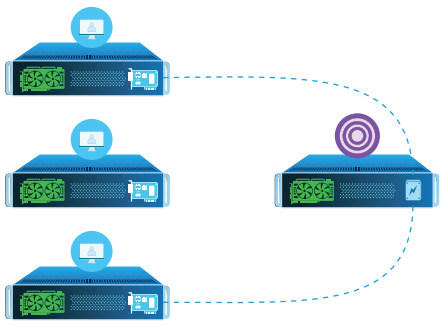
Another benefit of this architecture is the fact that NVMesh avoids noisy neighbors: in other software-defined storage systems, when you have multiple clients talking to shared storage, you are using CPU on the shared system in an unpredictable way. When you don't know when a host might place a very high IO load and tax the CPU, you have no predictability of the CPU load, which makes it very difficult to do establish quality of service. As **NVMesh does not use CPU**, it doesn't matter how active clients are against a target. This allows NVMesh to be deployed in a converged architecture with full application performance predictability.



So, if you have a rack full of servers and each server has an NVMe device, you don't need dedicated storage equipment, **which is another way NVMesh can save you money:** you get the highest levels of aggregate performance as well as the lowest cost overhead in capital acquisition by placing NVMe drives in each host. You may have been doing this already to accelerate your applications, but now you can share these drives amongst all these hosts, greatly improving overall utilization, without impacting the CPU of those hosts. Your rack of application servers can still do what you bought it for – to be application servers – but they also become shared storage servers, with no extra cost.

MAXIMUM UTILIZATION

NVMesh enables you to maximize performance and capacity utilization. If you have a rack of application servers and you decided to put NVMe in some or all of those servers to increase performance, you are making trade-offs. The first trade-off is capacity: if you put 4TB of NVMe in each server it means you can never use more than 4TB of space for an application in any one given server. The tendency is to pad estimates so you will likely get larger drives than you need: "let's buy 8TB drives, just in case". The problem though is that your overall capacity utilization in this scenario may be as low as 15 to 25% across all drives – just because you needed to make sure certain applications would have enough local storage. This is based on data we get from our web scale and enterprise customers. This is a very serious problem as NVMe is your most precious resource and you should be maximizing its utilization.



By sharing NVMe drives between application servers with NVMesh you can aggregate drives into larger virtual volumes (and you will get higher performance). So if you have multiple physical NVMe devices and each one is 4TB, you can put two devices together and have an 8TB virtual device (with RAID zero protection). You could also stripe across multiple systems to get more performance. For example, you can stripe across four devices but only take out 2TB slices. This gives you 8 TB but with the performance of four devices. Because we are a completely virtualized system, a virtual SAN, you can always resize the logical volume should you have created them too small.

HIGH PERFORMANCE

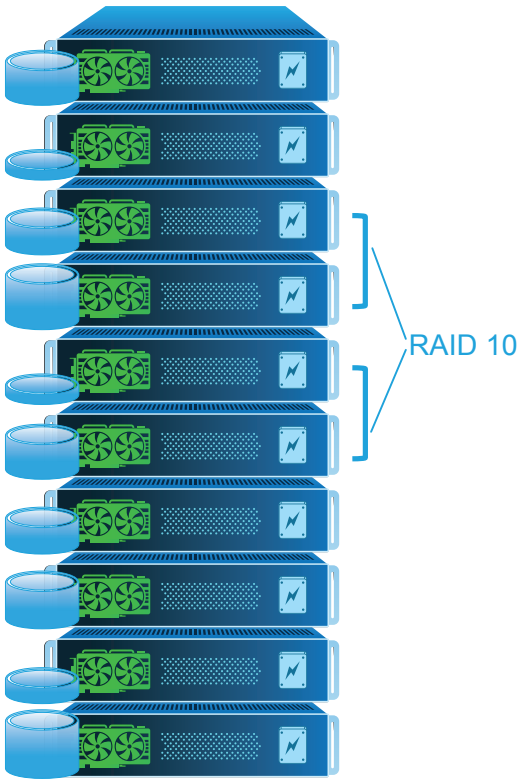
NVMesh®

NVMESH ENABLES YOU TO SHARE IOPS AND BANDWIDTH CAPACITY ACROSS SERVERS BY CREATING LOGICAL VOLUMES.

DATA PROTECTION

So, if you start out with 2TB and you need it to be 3TB you simply increase the size of the volume. Pooling your NVMe devices gives you the highest utilization levels as unused capacity can be distributed amongst all the logical volumes.

Additionally, you get very high performance utilization rates because you can share IOPs. You may have individual devices that can do 750,000 IOPs while your applications on a given server may only need 80,000 IOPs. Again, you have paid for an expensive resource that has an awful lot of performance capability that you're not tapping into. NVMesh enables you to share IOPs and bandwidth capacity across servers by creating logical volumes.

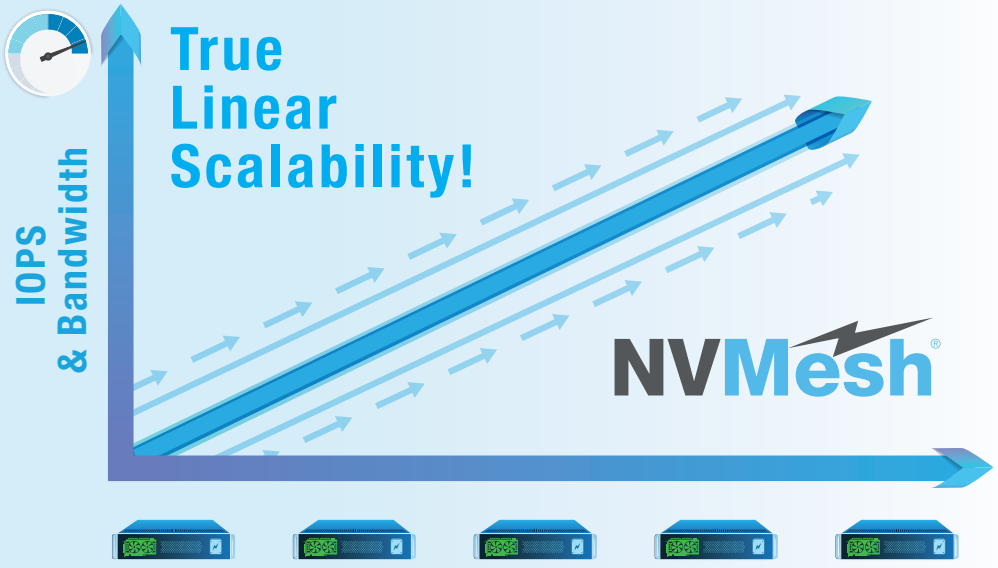


As mentioned before, NVMesh is an end-to-end software-defined storage solution that leverages NVMe. This means we also provide data protection. One option, for high performance is to mirror drives, for example in a RAID10, which does a striped mirroring over multiple devices. If there is a failure of a device or an entire host, your data remains available and protected. The logical volume will stay up. But we also offer erasure coding, which can offer higher protection with 60-80% usable capacity.

LINEAR SCALABILITY

NVMesh features 100% distributed block services and as such provides roughly 100% linear scalability. Let's take a look at how we achieve that. The graph below shows number of nodes on the horizontal axis and performance on the vertical axis. This is a very classical representation of storage performance. Most vendors will cut off the graph at a certain number of nodes because performance of their solutions will stop scaling at a certain cluster size, due to the architecture of the solution: past a certain number of nodes performance levels off due to inter-node communication – the protocols of the nodes speaking to each other within the cluster start to overwhelm the network, or their ability to do data processing.

For NVMesh that graph looks very different: as you add nodes, network bandwidth and NVMe drives, we increase our performance capability (IOPs and bandwidth). Because we are 100% distributed software-defined storage, with no inter-client communication and the very low overhead thanks to RDDA, we achieve near 100% linear scalability. There is no communication between the clients: clients are not talking to other clients and targets are not talking to other targets during normal operations. The data and control path are separate. If there is contention over a single block or a single NVMe device, those clients are resolving in that contention on the target server itself within that target host's memory. This is also why we have no centralized lock manager or metadata manager: our locking is 100% distributed and handled by the clients in the target side memory.



So **all these characteristics combined is what makes NVMesh revolutionary**: a 100% distributed system with near 100% efficiency and near 100% linear scalability.