Hewlett Packard Enterprise

Technical white paper

Check if the document is available in the language of your choice.

# HPE J2000 FLASH ENCLOSURES WITH LINUX RAID AND LVM

## High-level proof of concept

# CONTENTS



## EXECUTIVE SUMMARY

This white paper is a high-level proof of concept that describes how Linux® software RAID (mdraid) and Logical Volume Manager (LVM) can be used to provide resiliency to drive failure when used with the HPE J2000 Flash Enclosure. This investigation does not set out to reach the maximum performance possible but does illustrate how software can work with HPE J2000 Flash Enclosures and provides empirical data demonstrating the relative performance cost for a given configuration.

This paper does not provide detailed instructions on how to configure either Linux software RAID or LVM, both of which are documented by the currently supported Linux distributions of Red Hat® Enterprise Linux® 8, and SUSE Enterprise Server 15.

For guidance on how to configure the HPE J2000 with Linux, refer to the Implementing HPE J2000 Flash Enclosures with Linux Operating Systems white paper.

## INTENDED AUDIENCE

This white paper is written for systems administrators and architects who are investigating mechanisms to provide availability greater than the HPE J2000 Flash Enclosure or planned applications can provide natively. The reader should have working knowledge of:

- HPE J2000 Flash Enclosures
- NVMe-oF concepts
- Red Hat and SUSE Linux operating systems
- RAID technologies

## OVERVIEW OF THE CHALLENGE

The HPE J2000 Flash Enclosure provides extreme performance and consistently low latencies for the most demanding workloads. The system can be shared by multiple hosts that are either directly connected or fabric connected via a RoCEv2 capable switched infrastructure.

When configured optimally, the HPE J2000 Flash Enclosure can provide more than 10M IOPS and 70 GB/s of performance from a single 2U enclosure via twenty-four NVMe SSDs and six bridge cards[1]. There is no RAID, compression, deduplication or any other such service that would otherwise introduce unwanted latencies. Typically, if the application is unable to offer its own protection from failure and availability to data must be assured, traditional SAN array solutions are often an appropriate choice. However, SAN arrays always include some form of data services whereas the use of software protection solutions together with the HPE J2000 Flash Enclosure can protect application data while simultaneously providing extreme performance. If other traditional data services are required, then possibly a more traditional SAN array would be a more appropriate choice.

The HPE J2000 Flash Enclosure is a just a bunch of flash (JBOF) system that has all the performance benefits of internally located NVMe SSDs yet provides the flexibility of a fabric attached enclosure that can consolidate storage for multiple hosts and applications. This paves the way to server consolidation and enables swift reconfigurations of host infrastructure while remaining cost-effective. These advantages are discussed in detail in the Implementing HPE J2000 Flash Enclosures with Linux Operating Systems white paper.

## TESTING ENVIRONMENT AND METHODOLOGY

Because this paper is not intended to demonstrate the highest performance possible, a low-end server configuration was used with only a single 100GbE network port dedicated to data traffic. The average CPU usage during tests did not exceed 70% and averaged ~30%.

**IMPORTANT**
Although better performance is guaranteed when using more drives, more host ports, and multiples of faster CPUs, it is reasonable to conclude that performance will scale inline and relative to the results captured in this paper and within the limits of the HPE J2000 Flash Enclosure configuration. However, there are a multitude of variables such as application behavior and ability to multi-thread that prohibit the ability to provide any guarantees. Hewlett Packard Enterprise therefore highly recommends that an adequate test plan is formed in advance if performance is critical.

A single HPE J2000 Flash Enclosure with two ports was chosen with five NVMe SSDs. Each SSD was configured with a single 100 GB namespace and the host connected via a single HPE SN2700M switch. No other traffic was present on the network during testing.

[1] The HPE J2000 Flash Enclosure can be purchased with either two or six 100GbE data ports.

Multipathing was configured on the Linux hosts and both host ports on the HPE J2000 Flash Enclosure were used for every workload.

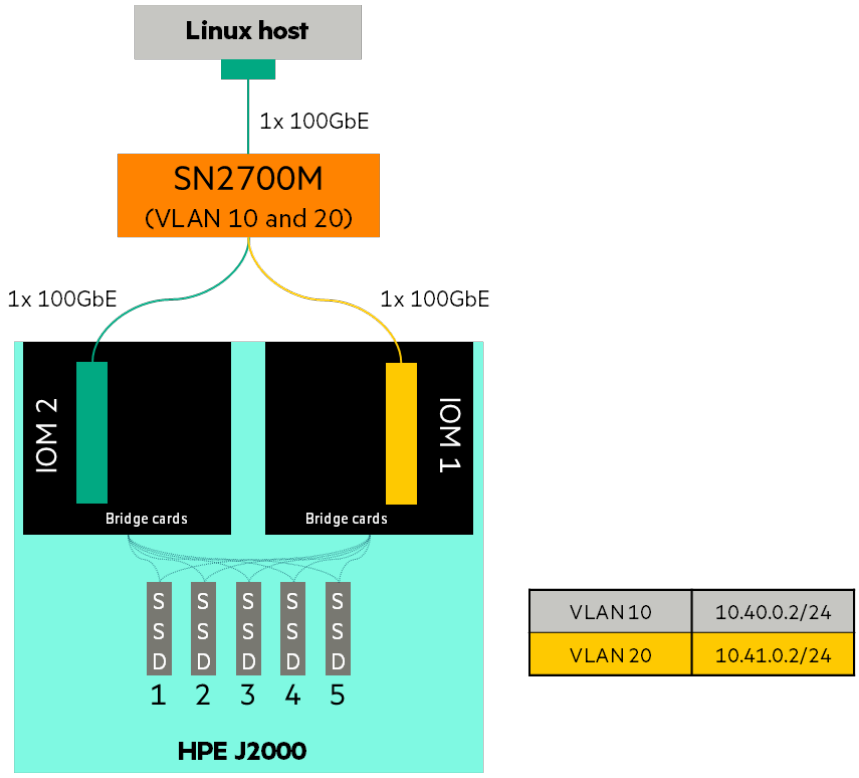


**FIGURE 1.** Environment used in this white paper

**TABLE 1.** Server configuration

| Component | Specification |
|---|---|
| **Server** | HPE ProLiant DL360 Gen10 |
| **Processor** | One Intel® Xeon®-Bronze 3104 1.7GHz |
| **Memory** | 16GB Dual Rank x8 DDR4-2666 CAS 19-19-19 RDIMM |
| **Data network** | HPE Ethernet 100Gb 1-port QSFP28 MCX515A-CCAT Adapter |
| **Operating system** | Red Hat Enterprise Linux 8 SP1 |

Application traffic was simulated using the freely available and commonly used Iometer benchmarking tool. To keep testing with all configurations fair, ten workers were used for all tests. "Worker" is an Iometer term that equates to an execution thread and is typically aligned to the number of virtual or physical cores within a server. For these tests, it was necessary to use additional threads due the requirement of a minimum of five storage devices to support RAID 6 with LVM, as well as a desire to use a minimum of two threads when addressing each drive natively.

All configurations underwent the same set of tests against the ext4 file system, and each worker addressed up to 104857600 blocks (50 GB). In each case the provisioning of both RAID and the file system completed before testing commenced.

## COMPARING PROTECTION TECHNOLOGIES

Both Red Hat and SUSE Linux enterprise distributions include the ability to implement software RAID and LVM at no additional cost. Linux software RAID and LVM offer different capabilities; the choice of which to use largely depends on application and requirements. Both offer the ability to protect data via various RAID schemes. The following RAID levels were tested with both random and sequential workloads and with both high and low queue depths:

- RAID 0/N-RAID
- RAID 10
- RAID 5
- RAID 6

**TABLE 2.** Comparison of Linux software RAID and LVM

| Solution | Summary |
|---|---|
| **Linux software RAID** | RAID is applied to the underlying storage device and creates a new RAID device onto which a file system can be applied. It offers protection but lacks flexibility. For example, you cannot partition the RAID device to apply multiple file systems and resizing a file system can be time-consuming because the RAID device must be resized first. |
| **LVM** | Each underlying storage device is grouped together to form a larger volume group. From that volume group, logical volumes of varying sizes can be created and different RAID protection schemes applied to each. Although more complicated to configure initially, logical volumes are more flexible and can be resized quickly. |

Testing was first conducted on file systems located on a native namespace (that is, no RAID). Theoretically, this configuration provides the best performance possible with the file system acting as the only software component in addition to Iometer. The expectation is that latencies will be the most consistent due to the lack of CPU overhead associated with RAID.

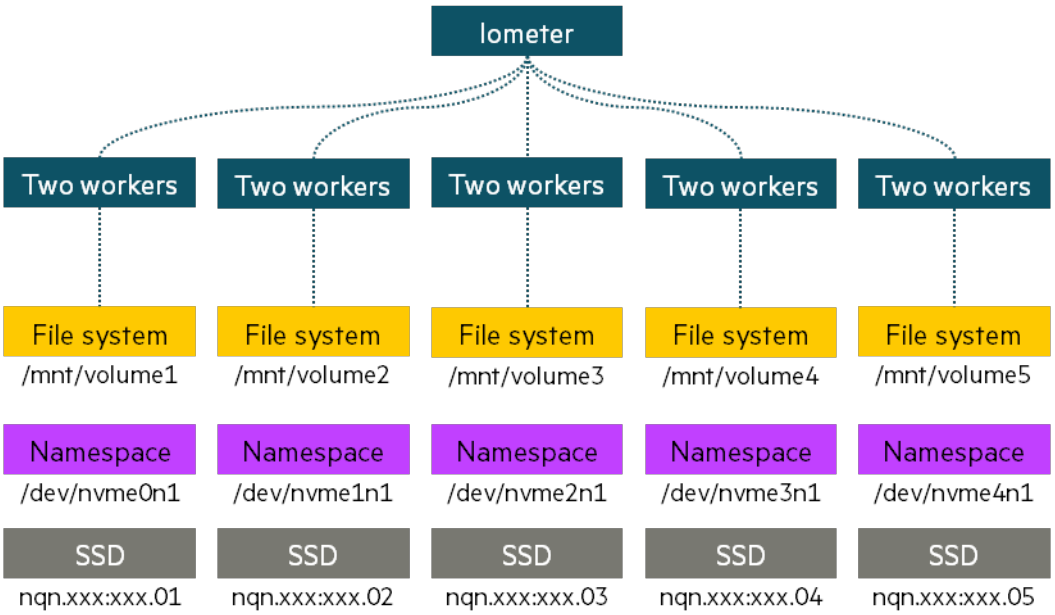


**FIGURE 2.** End-to-end device addressing when using native NVMe namespaces; no software RAID

When Linux software RAID is used, the entire capacity of each namespace is used to form a RAID device, and its entire capacity is used for the file system. This ensures that application traffic is evenly distributed across all drives while offering the ability to sustain drive failures.

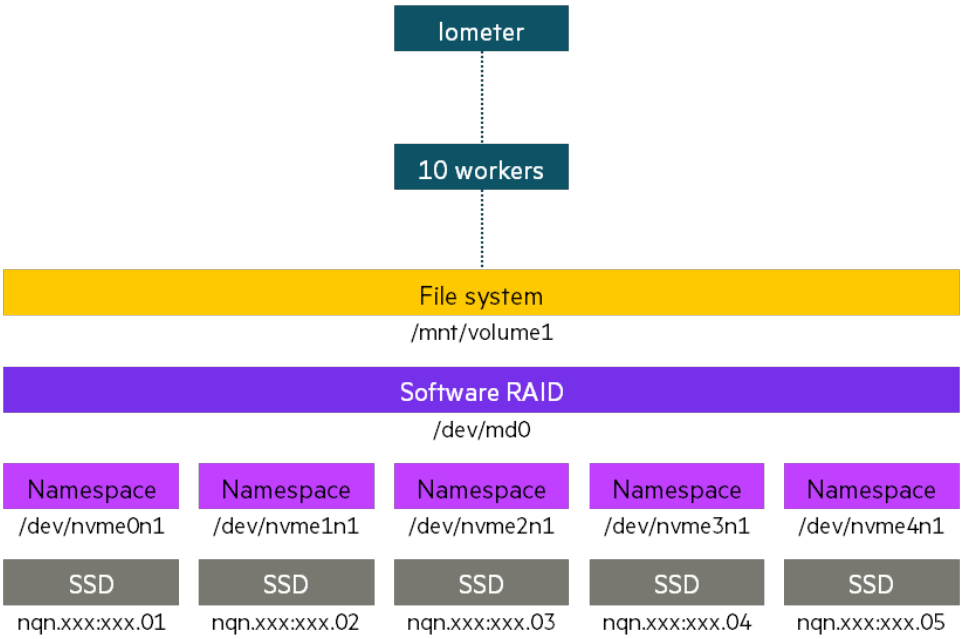


**FIGURE 3.** End-to-end device addressing when using Linux software RAID

LVM tests were conducted in the same manner except RAID was applied to the logical volume (lv1), which was configured to use all available capacity within the volume group shown in Figure 4. In such a configuration there is little differentiation between LVM and Linux software RAID although there is a provision for much more complex and varied volume management.

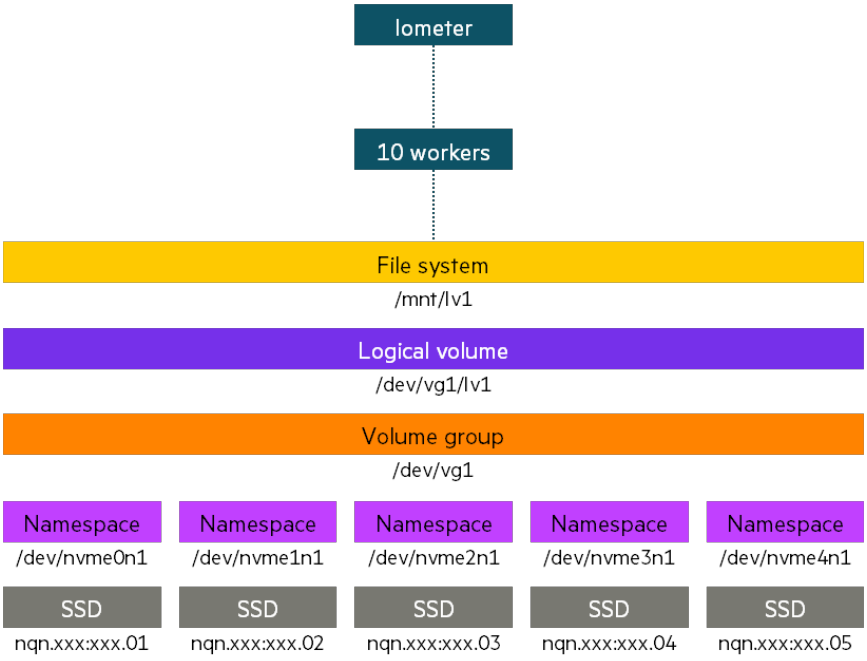


**FIGURE 4.** End-to-end device addressing when using LVM

## RESULTS

The following set of results is designed to illustrate the fundamental difference in performance across native drive access, Linux software RAID, and LVM, under both high and low queues depths and with the most common RAID types. Typically, higher queue depths result in better performance at the cost of latency, whereas low queue depths enable the lowest latencies at the cost of total performance.

Because testing showed a large differential in performance when using higher queue depths, comparisons between Linux software RAID and LVM were made using low queue depths. This decision was made so that a clear conclusion could be drawn and is appropriate when real-world performance variations derived from a low-end server configuration are accounted for.

Because native performance is plotted on the Linux software RAID graph, LVM testing made use of RAID 0 instead.

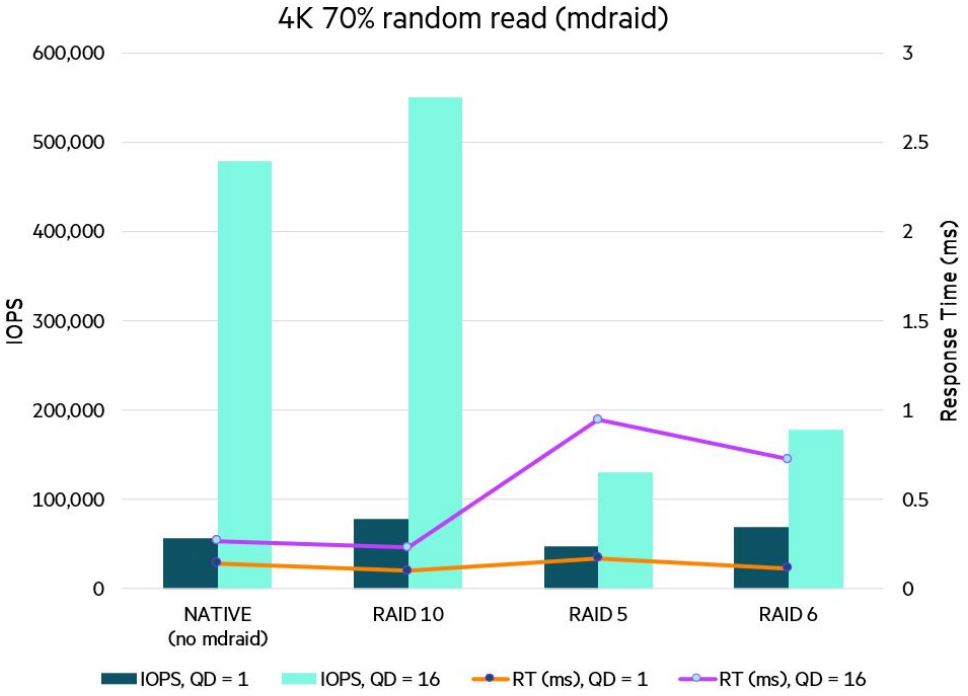


**FIGURE 5.** Mixed random performance of native namespaces compared with Linux software RAID

### Comments

As illustrated in Figure 5, these initial tests return a particularly interesting result when comparing native (no mdraid) access to Linux software RAID 10. What is ultimately striking is that performance increases when using software RAID and because this is the most likely RAID scheme to be used by most customers, is highly impressive. However, what is not possible to include in this graph is a plot of latency across time, which would reveal that although the average latency of Linux software RAID was low, it was not perfectly consistent. The lower performance when using no RAID serves to demonstrate the variations that can be expected when using a limited configuration on low-end server hardware.

Furthermore, although parity-based RAID showed significant reduction in performance, it still shows impressive performance from limited hardware. Again, hardware limitations expose unexpected performance gains for the more taxing RAID 6 over RAID 5. Yet in all cases the response times were on average extremely low. For many workloads, especially at scale, all configurations are likely acceptable.

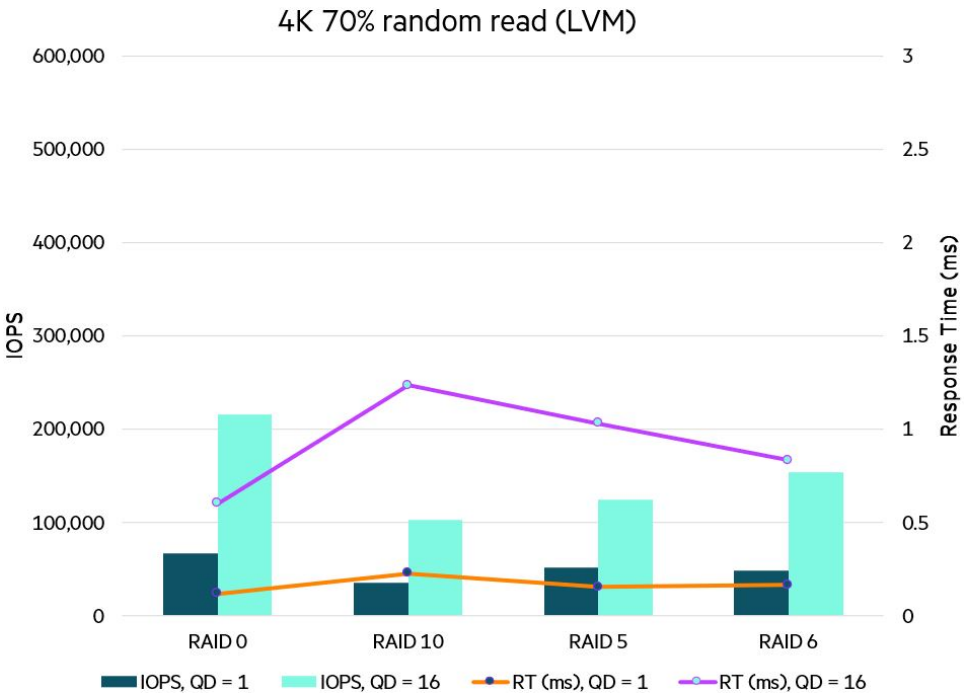


**FIGURE 6.** Mixed random performance of RAID 0 compared with other RAID schemes of LVM

### Comments

The graph in Figure 6 shows LVM performance with a higher queue depth is markedly lower for RAID 0 and RAID 10 than that of Linux software RAID, although it is relatively consistent across all RAID schemes.

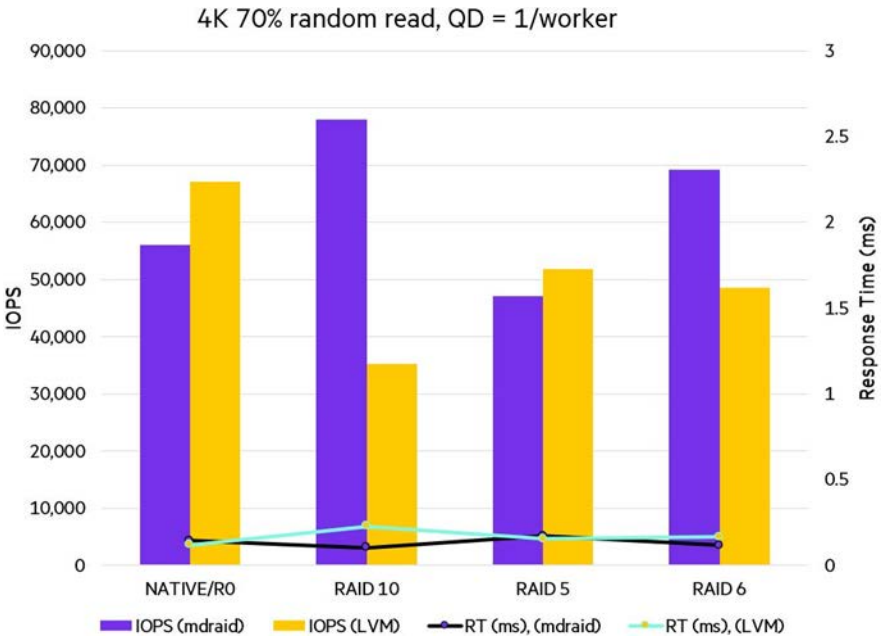


**FIGURE 7.** Mixed random performance comparison of Linux software RAID and LVM when using a low queue depth

### Comments

Figure 7 is a representation of performance with low queue depth and indicates that Linux software RAID, LVM, and native access (N-RAID) all performed very similarly and with extremely low response times. It is reasonable to deduce that for this configuration, this is the minimum performance that would be observed.

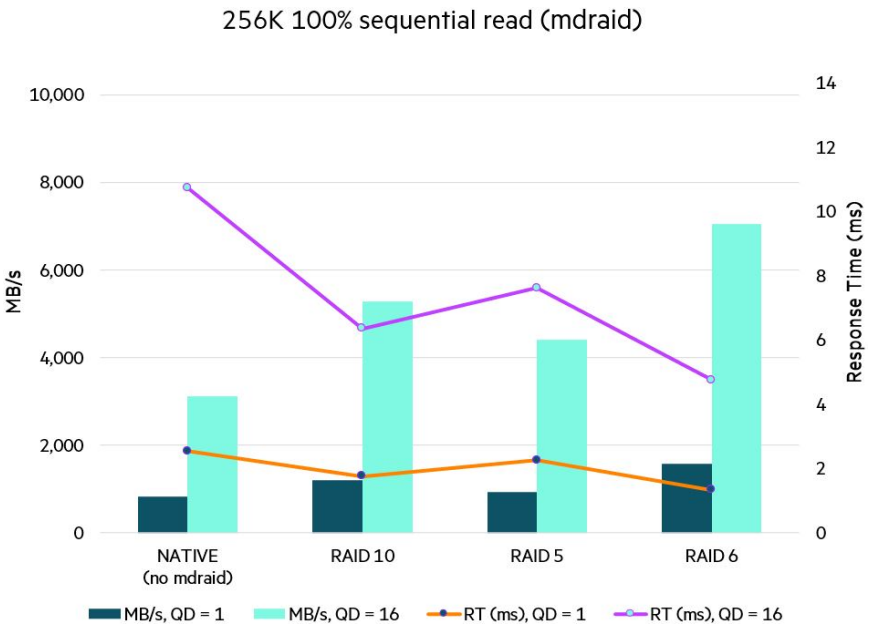


**FIGURE 8.** Sequential read performance of native namespaces compared with Linux software RAID

### Comments

Figure 8 shows Linux software RAID again has performed extremely well, especially with RAID 6. Latency has crept up yet is still not unacceptably high with a higher queue depth. It is most likely directly tied to the low-end CPU. Again however, with low queue depths, Linux software RAID has proven that it can deliver impressive performance.

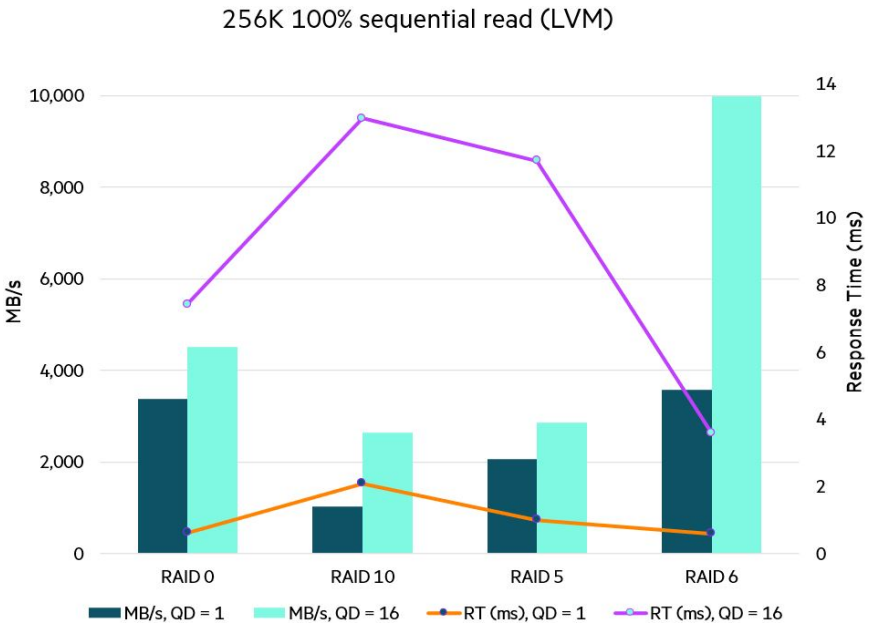


**FIGURE 9.** Sequential read performance of RAID 0 compared with other RAID schemes of LVM

### Comments

Figure 9 shows RAID 0 to be a good configuration for dominantly sequential reads but is an unlikely configuration for most applications because it introduces multiple points of failure. In other RAID configurations, LVM also performed well but performed extremely well in RAID 6, delivering nearly 10 GB/s of throughput. However, it is highly likely that the single 100 Gb connection from the server introduced a bottleneck.

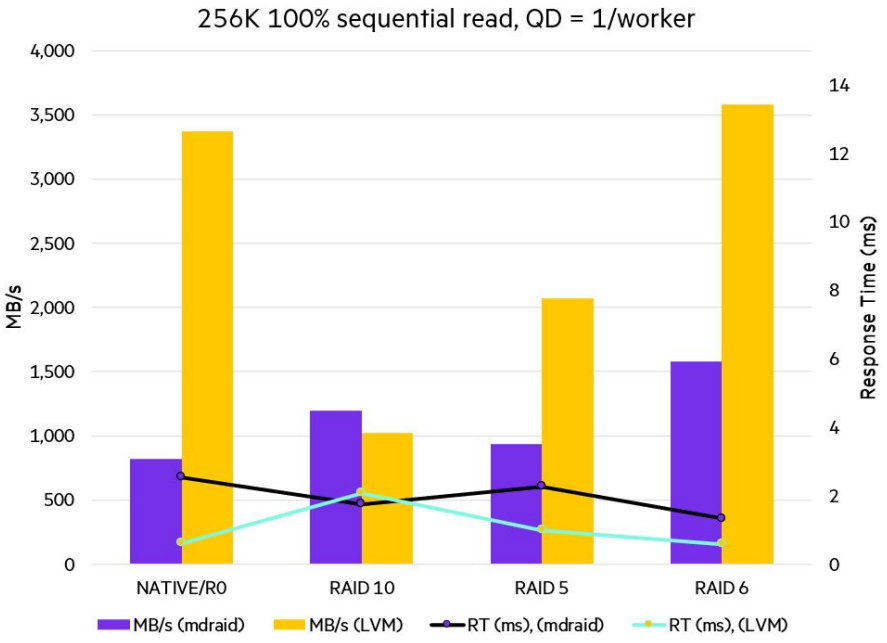


**FIGURE 10.** Sequential read performance comparison of Linux software RAID and LVM when using a low queue depth

### Comments

Figure 10 shows that with higher queue depths, both Linux software RAID and LVM perform quite similarly. However, LVM is a clear winner for RAID 6 performance under dominantly sequential read workloads regardless of queue depth.

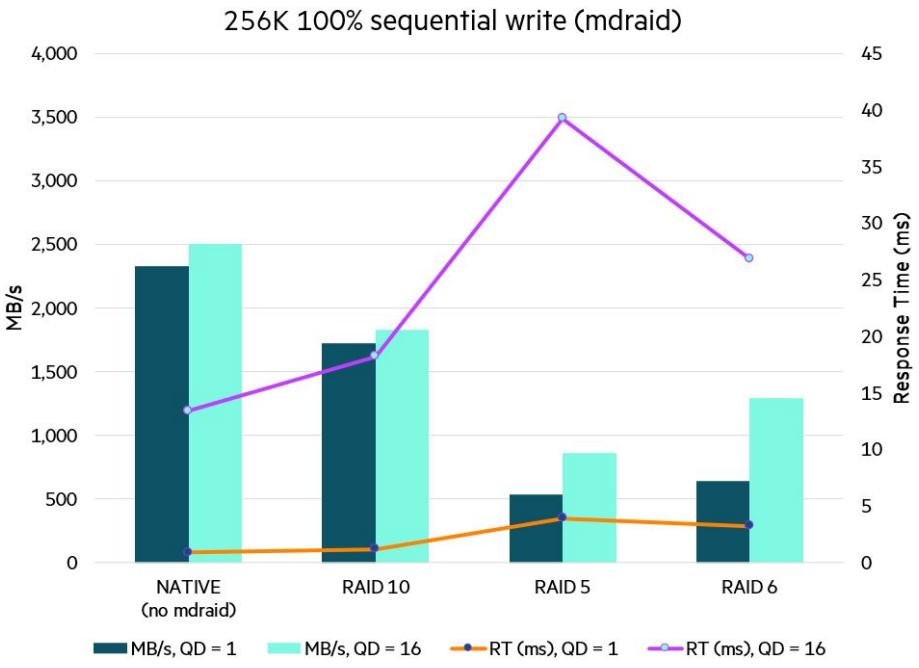


**FIGURE 11.** Sequential write performance of native namespaces compare with Linux software RAID

### Comments

Sequential write performance when using Linux software RAID, shown in Figure 11, does exhibit reduced performance when compared to native drive access, but response times remain low on average. This is the inverse of measured random read performance, which for RAID 10 was higher, yet more in keeping with expectations.

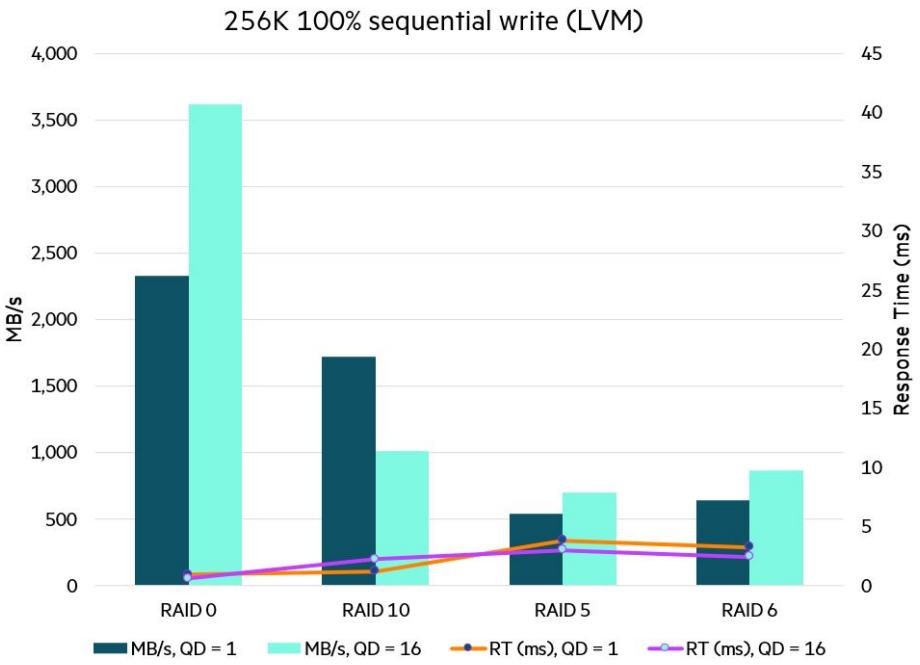


**FIGURE 12.** Sequential write performance of RAID 0 vs other RAID schemes of LVM.

### Comments

LVM produces impressive results for raw throughput, shown in Figure 12. The most impressive data is the consistently low average response times regardless of queue depth.

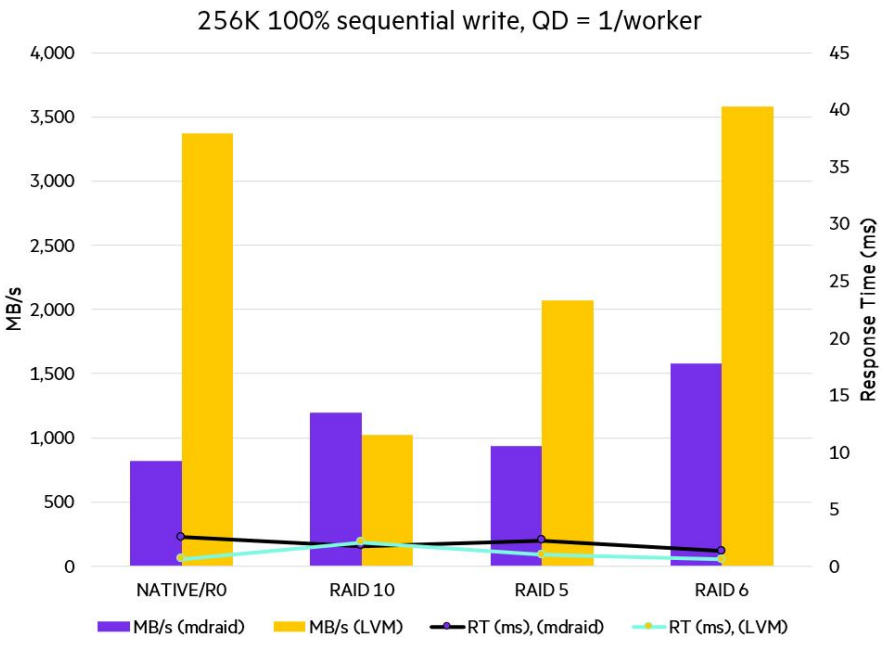


**FIGURE 13.** Sequential write performance comparison of Linux software RAID and LVM when using a low queue depth.

### Comments

In this configuration, Figure 13 shows LVM as a clear leader regarding sequential write performance compared with both native namespace access and Linux software RAID.

## CONCLUSION

It is clear from testing that the server hardware and network configuration used in these tests can lead to lower and less consistent performance than what could be achieved when using multiple well provisioned hosts, a six-port HPE J2000 Flash Enclosure, more drives, and more host ports. However, what is also clear is that the use of software solutions to provide resiliency to drive failure is both plausible and in some cases, beneficial to performance.

Linux software RAID, although less flexible, can provide impressive mixed random performance, especially when configured in RAID 10. Though its sequential performance is on average lower than that of LVM, it may be that for certain applications it is more than sufficient. Conversely, LVM offers greater flexibility and superior sequential performance, even over native access during these tests.

What is most striking is that average latencies can reach the levels of native access, which is highly impressive. Sequential write workloads using LVM are of particular interest due to its indifference to queue depth. Again, the consistency of latency will vary as will all measurable performance when there are multiple workloads and different storage and server configurations.

Although not all workloads require resiliency, it is proven that not only are both Linux software RAID and LVM able to work with the HPE J2000 Flash Enclosure, but it may also be the best choice. Critically, because it is possible to divide SSDs into multiple namespaces and namespaces can be presented to separate servers and applications, it is not necessary to make a single decision. Instead, it is possible to apply one method to some applications and another method to others. This level of flexibility coupled with the uncompromising performance of the HPE J2000 Flash Enclosure positions it as a perfect solution for the most demanding environments.

**Resources, contacts, or additional links**

Implementing HPE J2000 Flash Enclosure with Linux Operating Systems
https://www.hpe.com/psnow/doc/a00111247enw?jumpid=in_lit-psnow-red

J2000 Flash Enclosure QuickSpecs
https://www.hpe.com/psnow/doc/a00094645enw?jumpid=in_lit-psnow-red

HPE J2000 Documentation
https://www.hpe.com/support/J2000

HPE J2000 Firmware
https://www.hpe.com/storage/DiskEnclosureFirmware

HPE Product & Solutions Now
https://psnow.ext.hpe.com/

## LEARN MORE AT

hpe.com/storage



a00119659ENW

Hewlett Packard Enterprise