December 2022

Non-Volatile Memory Express Performance with Linux on IBM® LinuxONE Emperor 4

Results of use cases

82037982-USEN-01

Contents

List of Figures

1 About this publication

This white paper examines the setup, tuning, and performance aspects of direct attached Non-Volatile Memory Express (NVMe) storage on IBM® LinuxONE Emperor 4 using Linux® on IBM Z®.

1.1 Authors

IBM Systems Performance Linux on IBM Z team:

- Dr. Dominic Roehm
- Ebru Zengin
- Dr. Muhammad Usman Karim Khan
- Dr. Stefan Wirag
- Wolfgang Taphorn

IBM Linux on IBM Z Performance Evaluation team:

- Martin Kammerer

1.2 Remarks

The web-links referred in this paper are up to date as of October 2022.

1.3 Tested SSDs from earlier document releases

The following NVMe SSDs have been tested by IBM on an IBM® LinuxONE II:

- Intel SSDPE2KX010T801 (1TB)
- Intel SSDPE2KX040T801 (4TB)

The following NVMe SSDs have been tested by IBM on an IBM® LinuxONE III:

- Intel SSDPE2KX010T801 (1TB)
- Intel SSDPE2KX040T801 (4TB)
- Samsung MZWLL12THMLA-00005 (12.8TB)

The following NVMe SSDs have been tested by IBM on an IBM® LinuxONE Emperor 4:

- Kioxia CM6-V KCM6XVUL1T60 (1.6TB)
- Kioxia CM6-R KCM6XRUL1T92 (1.92TB)
- Kioxia CM6-R KCM61RUL30T7.5 (30.72TB)
- Samsung PM1733 MZWLJ1T9HBJR-00007 (1.92TB)
- Samsung PM1733 MZWLJ15THALA-00007 (15.36TB)

2 Introduction

Introductory information is provided in this section about the setup, tuning, and performance aspects of direct attached Non-Volatile Memory Express (NVMe) storage on IBM® LinuxONE Emperor 4 using Linux on IBM Z. More information about IBM® LinuxONE Emperor 4 can be found at:

- https://www.ibm.com/it-infrastructure/linuxone
- https://www.ibm.com/products/linuxone-emperor-4

The latest NVMe specifications can be found at:

- https://nvmexpress.org/resources/specifications

2.1 Objectives of this white paper

The high-performance characteristics of NVMe storage have the potential to improve response times and throughput of I/O intense use case scenarios.

The expectation is that NVMe storage provides fast and inexpensive storage for servers using sufficient NVMe storage capacity to contain the workload data. This includes workload data for applications such as MongoDB®, ScyllaDB®, Apache® Cassandra®, IBM® Db2® Warehouse, Apache Spark™, Apache Hadoop®, PostgreSQL® or MariaDB®, and Linux containers along with KVM host/guests.

Other use-cases of NVMe storage could be non persistent data like IBM Db2 Warehouse external sort temp space, Oracle Database cache devices, Apache Hadoop scratch disk as well as paging devices for hypervisor hosts.

This paper demonstrates the performance using NVMe storage with a set of performance comparisons for two selected use case scenarios:

- PostgreSQL database workload
- IBM Db2 Warehouse database workload

Performance criteria of I/O throughput metrics are compared when running the workloads and the obtained results are used to calculate price-performance numbers. Individual measurements are conducted for each use case scenario.

2.2 Summary

The performance evaluation of databases using direct-attached NVMe SSD storage versus external storage shows that it depends on the database disk I/O characteristic and the number of connections to the external storage whether the workload performance improves using NVMe storage or not.

Using four direct-attached NVMe SSDs in RAID-10 setup versus four FCP attached LUNs connected through two FICON Express32S adapters to an IBM DS8950F server with flash card set storage in RAID-0 results in a similar transactions per second rate (throughput) using the YCSB benchmark connected to a PostgreSQL 14.1 running on IBM® LinuxONE Emperor 4. Since the YCSB benchmark is emitting a continuous stream of queries against the database, the PostgreSQL database also creates continuous disk I/O requests, which are processed by the I/O subsystem and the storage in parallel. Hence, PostgreSQL throughput depends upon the number of I/O requests that the storage can process at a time and therefore the number of direct-attached NVMe SSDs and the used RAID setup for those versus the number of used I/O adapters and connections to the storage server matters. To conclude, we cannot state that direct-attached NVMe SSDs always deliver a performance advantage compared to external storage and vice versa for a workload with a steady disk I/O.

Measurements using IBM Db2 Warehouse used the Big-Data-Insights (BDI) benchmark, which executes a broad set of OLAP queries with varying complexity and measures their execution time. The measured throughput (Queries per Hour) is similar using four direct-attached NVMe SSDs in a RAID-10 setup compared to using four FCP attached LUNs connected through two FICON Express32S adapters to an IBM DS8950F server with flash card set storage in RAID-0.

The price-performance evaluation for the PostgreSQL and the IBM Db2 Warehouse scenarios shows that direct-attached NVMe SSDs have a compelling price-performance advantage when comparing the purchase costs of NVMe SSDs with the IBM NVMe Carrier versus the purchase costs of an external storage servers with the same amount of raw storage capacity.

For the PostgreSQL scenario the price-performance advantage of direct-attached NVMe SSD storage is up to 96% versus FCP attached LUNs connected through two FICON Express32S adapters to an IBM DS8950F server with flash card set storage.

For the IBM Db2 Warehouse scenario the price-performance advantage of direct-attached NVMe SSD storage is up to 94% versus FCP attached LUNs connected through two FICON Express32S adapters to an IBM DS8950F server with flash card set storage.

2.3 NVMe Protocol Overview

The NVMe protocol is an open collection of standards and interfaces that fully exposes the benefits of non-volatile memory in all types of computing environments, from mobile to data center. It is designed to deliver high bandwidth and low latency storage access.

IBM has a long history of commitment to industry standards, and the NVM Express organization is no exception. As a member of the NVMe workgroup, IBM aims to contribute to the standards and to lead the enablement of enterprise storage systems that use the new drives, network protocols, and I/O architectures.

The NVM Express consortium develops the NVM Express specification and describes how NVMe differentiates itself from other interfaces, such as SATA and SAS, as follows:

- NVMe is an optimized, high-performance scalable host controller interface designed to address the needs of Enterprise and Client systems that utilize PCI Express-based solid-state storage. Designed to move beyond the classical hard disk drive technology, NVMe is built from the ground up for non-volatile memory technologies. NVMe is designed to provide efficient access to storage devices built with non-volatile memory, from today's NAND flash technology to future, higher-performing, persistent memory technology
- There are several performance vectors that NVMe addresses, including bandwidth, IOPs, and latency. For example, the maximum IOPs possible for Serial ATA was 200,000, whereas NVMe SSD devices have already been demonstrated to exceed 1,000,000 IOPs. By supporting PCI Express and Fabrics such as RDMA and Fibre Channel, NVM Express can support much higher bandwidths than SATA or SAS (e.g., a PCI Express Gen3 $\times$4 delivers 4 GB/s). Finally, next generation memory technologies may have read access latency under a microsecond, requiring a streamlined protocol that enables an end-to-end latency of under10 microseconds, including the software stack.

2.4 Linux disk I/O general flow description

For Linux NVMe storage adds an alternative way of executing disk I/O compared to the existing FICON/ECKD and FCP/SCSI protocols. “Figure 1 Overview of Linux disk I/O flow” shows an overview of Linux disk I/O flow.

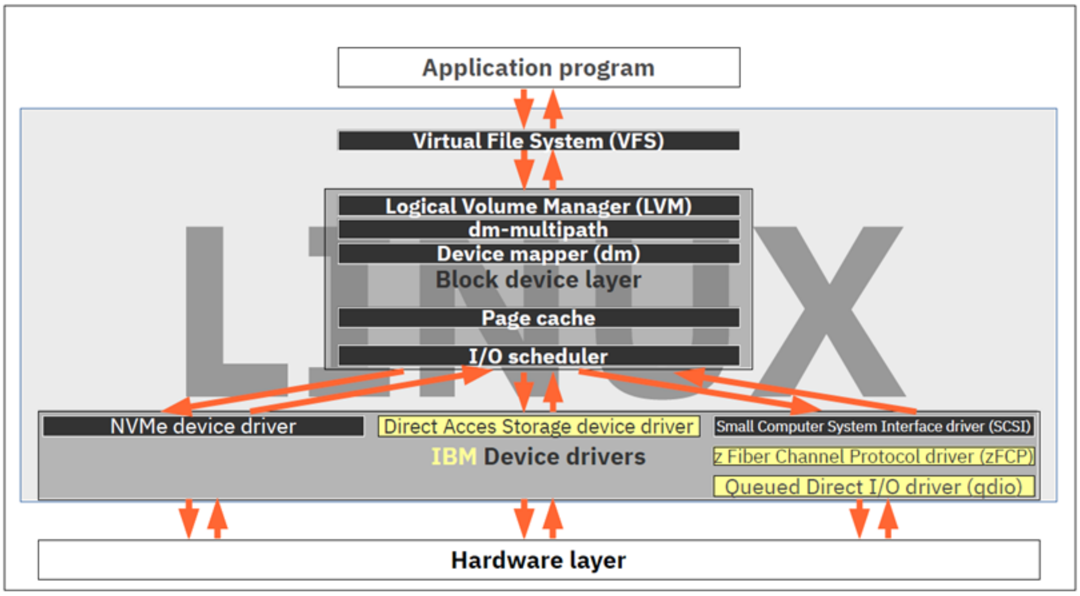

Figure 1: Overview of Linux disk I/O flow

Disk I/O in Linux is processed in these step

- The application program is issuing reads and writes (system calls) and the Linux block device layer is entered
- The Virtual File System (VFS) dispatches requests to different devices and translates the data buffers to sector addressing
- The Logical Volume Manager (LVM) is used to define the physical to logical device relation
- Multipath sets the multipath policies
- The device mapper (dm) holds the generic mapping of all block devices and performs 1:n mapping for logical volumes and/or multipath devices
- The page cache contains all file I/O data
- Direct I/O bypasses the page cache
- The I/O schedulers merge, order, and queue requests, start device drivers via unplug/plug device
- The appropriate device driver communicated with the hardware layer depending on the device type
- Direct Access Storage Devices (DASD) are handled by the Linux specific Linux device driver, using the FICON/ECKD protocol
- Small Computer System Interface (SCSI) devices are handled by the Linux common code SCSI device driver, followed by the Linux on IBM Z specific Z FiberChannel Protocol (zFCP) device driver for wrapping SCSI with FCP and Queued Direct I/O (qdio) for transferring the data to the FICON Express card, using the FCP/SCSI protocol
- NVMe SSD devices are handled by the Linux common code NVMe device driver, using the NVMe protocol

3 Description of the system under test (SUT)

The SUT was based on a Linux system running in an LPAR with access to direct-attached NVMe SSD storage, FCP attached LUNs on an IBM DS8950F server with flash card set storage and network connections to another LPAR hosting workload drivers.

Note: The selection and purchase of the SSDs used in this feature is the responsibility of the client. IBM can only comment on the reliability and performance of SSDs that have been tested. The client assumes all risk in, and IBM is not responsible for the use of SSDs as the functionality and/or performance may vary.

The following NVMe Solid State Drives (SSDs) have been tested by IBM on IBM® LinuxONE Emperor 4:

- Kioxia CM6-V KCM6XVUL1T60 (1.6TB)
- Kioxia CM6-R KCM6XRUL1T92 (1.92TB)
- Kioxia CM6-R KCM61RUL30T7.5 (30.72TB)
- Samsung PM1733 MZWLJ1T9HBJR-00007 (1.92TB)
- Samsung PM1733 MZWLJ15THALA-00007 (15.36TB)

In our case studies we used 4x Samsung PM1733 NVMe SSDs of 1.92TB capacity, each plugged into an IBM NVMe Carrier (Feature Code=0448). For simplification in this document, those are referred to as 1.92TB NVMe SSDs.

NVMe support is included in all recent SUSE, Red Hat, and Ubuntu distributions for IBM® LinuxONE. The minimum Linux distributions for IBM® LinuxONE, which include NVMe support are:

- SLES 12 SP4, SLES 15
- RHEL 7.6 + z-stream, RHEL 8.0
- Ubuntu 17.10

NVMe SSD storage serves as block device for all kind of data including dump and swap. Since Linux kernel v5.8, is it possible to IPL Linux from NVMe SSD storage on native LPAR and KVM:

- SLES 15 SP3
- RHEL 8.3, RHEL 9.0
- Ubuntu 20.04

3.1 IBM® LinuxONE Server

The case study involved the configuration of LPARs in an IBM® LinuxONE Emperor 4, Model 3931-LA1.

To determine if the NVMe SSD devices are recognized by Linux, run the following command:

```
$> lspci
000b:00:00.0 Non-Volatile memory controller: Samsung Electronics Co Ltd NVMe
SSD Controller 172Xa/172Xb (rev 01)
000e:00:00.0 Non-Volatile memory controller: Samsung Electronics Co Ltd NVMe
SSD Controller 172Xa/172Xb (rev 01)
0010:00:00.0 Non-Volatile memory controller: Samsung Electronics Co Ltd NVMe
SSD Controller 172Xa/172Xb (rev 01)
0013:00:00.0 Non-Volatile memory controller: Samsung Electronics Co Ltd NVMe
SSD Controller 172Xa/172Xb (rev 01)
```

Figure 2: NVMe drives are recognized on this system and can be used

Afterwards the $parted$ command is used in Linux to create partitions. Once the partitions are created, one can use the NVMe SSDs as block storage devices, or use software-based RAID to create a RAID configuration. Below, we show an excerpt of creating a RAID-10 array based upon the 4 NVMe SSD drives we have on the SUT:

```
$> apt install mdadm
$> mdadm --create /dev/md0 --chunk=256 --level=10 --raid-devices=4 \
     /dev/nvme0n1p1 /dev/nvme1n1p1 /dev/nvme2n1p1 /dev/nvme3n1p1 --verbose
$> mkfs.xfs /dev/md0
$> mount /dev/md0 /nvmeData
```

Figure 3: First install mdadm and then create RAID-10 based upon the 4 NVMe drives

Now, the mounts can be verified with the usual commands, for example:

```
$> lsblk
NAME     MAJ:MIN RM  SIZE RO TYPE   MOUNTPOINT
nvme2n1  259:5    0  1.8T  0 disk
└─md127    9:127  0  3.5T  0 raid10 /nvmeData
nvme1n1  259:6    0  1.8T  0 disk
└─md127    9:127  0  3.5T  0 raid10 /nvmeData
nvme0n1  259:7    0  1.8T  0 disk
nvme3n1  259:8    0  1.8T  0 disk
└─md127    9:127  0  3.5T  0 raid10 /nvmeData
nvme4n1  259:9    0  1.8T  0 disk
└─md127    9:127  0  3.5T  0 raid10 /nvmeData
```

Figure 4: 4× 1.9TB devices verified by the output of the lsblk command

3.2 PostgreSQL test

3.2.1 PostgreSQL LPAR setup

The LPAR setup running the PostgreSQL server consisted of:

- 30 dedicated cores
- 16 GB storage
- 1× 25GbE RoCE Express2 card
- 2× FICON Express32S cards, using the FCP/SCSI protocol
- 4× 3.8TB LUNs on DS8950F flash card set storage server
- 4× IBM NVMe Carrier
- 4× 1.92TB NVMe SSDs

The software used was:

- PostgreSQL 14.1
- Ubuntu 20.04.3 LTS Server Edition

3.2.2 YCSB benchmark LPAR setup

The LPAR setup running the YCSB workload driver consisted of:

- 16 dedicated cores
- 64 GB storage
- 1× 25GbE RoCE Express2 card

The software used was:

- Ubuntu 20.04.3 LTS Server Edition
- Yahoo! Cloud Serving Benchmark (YCSB) 0.17.0

3.3 IBM Db2 Warehouse test

3.3.1 IBM Db2 Warehouse LPAR setup

The LPAR setup consisted of:

- 30 dedicated core processors
- 128 GB or 256 GB storage
- 2× FICON Express32S cards, using the FCP/SCSI protocol
- 4× 3.8TB LUNs on DS8950F flash card set storage server
- 4× IBM NVMe Carrier
- 4× 1.92TB NVMe SSDs

The software used was:

- Ubuntu 20.04.3 LTS Server Edition
- IBM Db2 Warehouse 11.5.7.0-cn3
- OpenJDK Runtime Environment 11.0.15
- IBM internal Big Data Insights (BDI) benchmark

3.4 PostgreSQL performance test

The purpose of this test is to determine the performance of the PostgreSQL database server under heavy transactional workload. The PostgreSQL server stores the data on either the NVMe SSDs, or on the IBM DS8950F server with flash card set storage. A transactional workload driver sends a request to the database server to search for a particular row/record within a table using an ID or key of the record. This request can either be a read request to send data back to the driver or an update/write request to write data to the table.

When a read-request is submitted to the server, the row/record is searched if it resides in the memory. If not, then the data must be fetched from the storage system, in our case from the NVMe SSDs or from the FCP attached LUNs. Moreover, if the memory size increases, the I/O traffic decreases and vice versa.

The YCSB program suite was used to drive the workload in both scenarios from another LPAR using a 25 Gbps network.

The scenarios are described by these parameters:

- Number of users (= threads) fixed to 256
- Record size: 64 KB
- FCP attached LUNs connected through 2 FICON Express32S adapters and configured as RAID-0
- NVMe SSDs configured as RAID-10
- The throughput is measured in transactions per second
- The PostgreSQL database size was 256 GB
- The measurements ran for 30 minutes
- Experiment 1: Read-Only (95% read)

- Experiment 2: Read-Mostly workload (95% read, 5% update)
- Experiment 3: Update-Heavy workload (50% read, 50% update)
- Every test first has three stages: first load the database, then run a warm up phase, and finally run the above described experiments

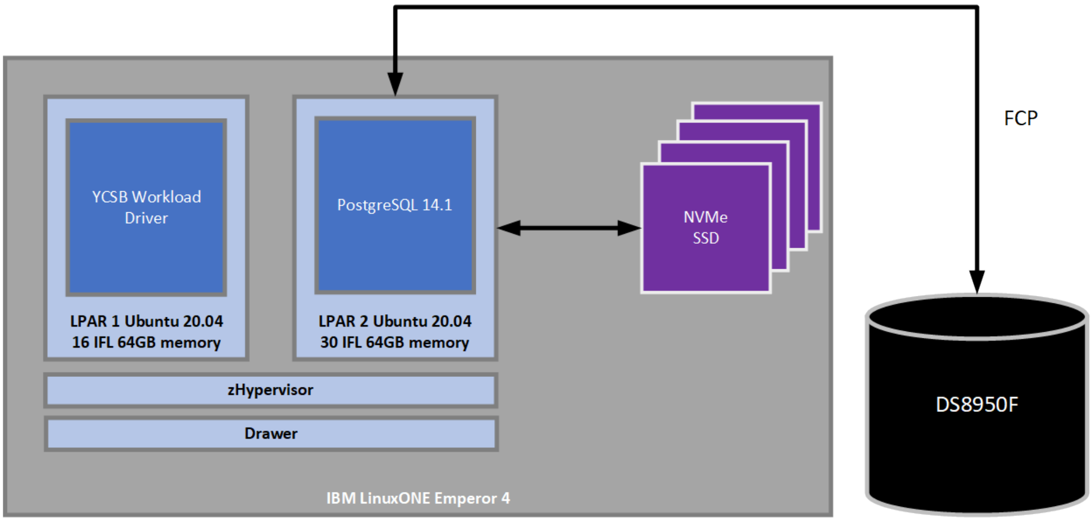

Figure 5: PostgreSQL benchmark configuration showing IBM DS8950F server with flash card set storage and NVMe attachment

3.5 IBM Db2 Warehouse performance test

In comparison to a transactional workload, an analytical workload will search for a large set of data within the database and then return the result to the workload driver. The workload driver sends a "query" to the database server. A query is a composition of multiple conditions, and the database returns the set of row/records to the driver, which satisfies these conditions. The complexity of the query determines the amount of work the database server must process. If the size of the database is larger than the memory size, the database server must search for the rows/records that satisfy the query conditions on the external disks and thus cause I/O traffic. The purpose of these tests is to determine the impact of the external storage on the performance of such queries (i.e., time in finding all the rows/records satisfying the conditions). For these scenarios, the database either resides on the NVMe SSDs or the IBM DS8950F server with flash card set storage.

We used Big-Data-Insights (BDI) benchmark to test the performance of a server running IBM Db2 Warehouse. BDI is executing a broad set of OLAP queries with varying complexity. In these tests, there are multiple parallel drivers, and at one time each pushes a single query to the IBM Db2 Warehouse database. The drivers fetch these queries from a set, with queries categorized as simple, intermediate, and complex queries. Thus, there is a mixture of simple, intermediate, and complex queries running simultaneously on the database server. These queries will cause I/O traffic, and the latency of a read or write request to the external drives would determine the performance of the database server.

The workload drivers resided for this set of tests in the same LPAR since it has a very low resource consumption.

The scenarios are described by these parameters:

- Number of parallel users fixed to 6, performing predefined SQL queries
- Record size: average 6 KB
- The throughput is measured in Queries per Hour
- FCP attached LUNs connected through 2 FICON Express32S adapters and configured as RAID-0
- NVMe SSD devices configured as RAID-10

- Scaling Linux memory for 128 GB and 256 GB
- The IBM Db2 Warehouse database size was 500 GB

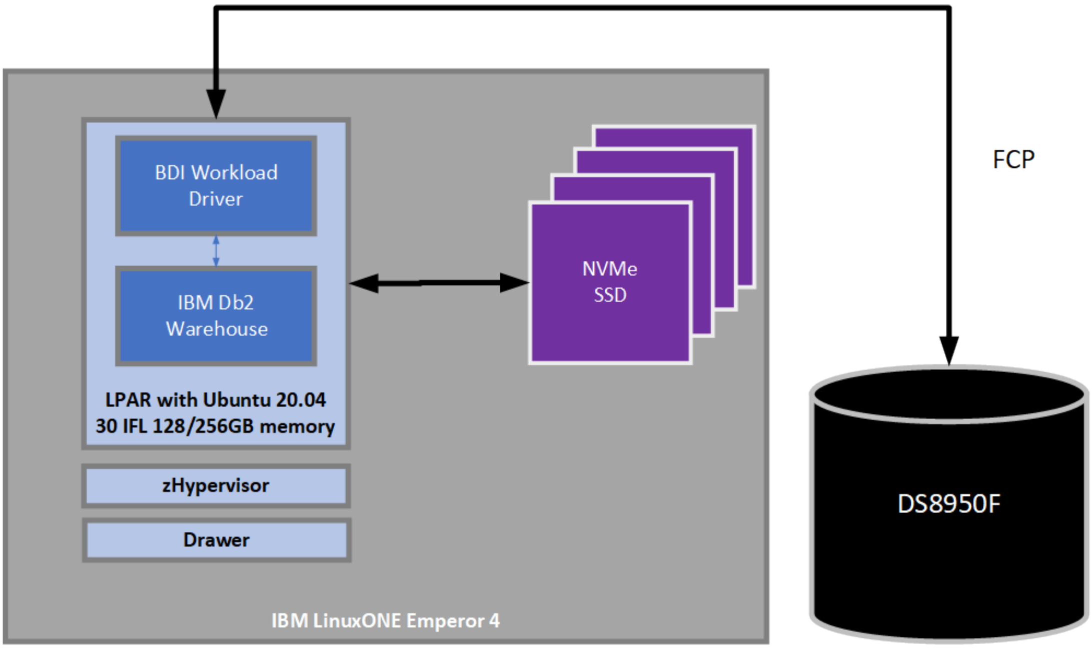

Figure 6: IBM Db2 Warehouse benchmark configuration showing IBM DS8950F server with flash card set storage and NVMe attachment

4 Case study results

For the measurement series discussed in this chapter all the described configurations were applied to the SUT.

4.1 Transaction throughput

This section shows the throughput rate for the executed workloads on the SUT.

4.1.1 Observations

The throughput rate when running the YCSB benchmark on PostgreSQL 14.1 shows an advantage of up to 16% for the local NVMe devices for the three use-cases. The overall read/write rates are 90% read and 10% write at a very high and constant read rate >3 GB/s. The local NVMe RAID-10 setup has a theoretical read bandwidth of almost 10 GB/s. The throughput in our specific configuration via the four FCP attached LUNs on an IBM DS8950F server with flash card set storage with the underlying RAID-0 is saturating around 3 GB/s. This is the reason for the advantage of the local NVMe RAID setup observed in Figure 7. The I/O limits observed in this particular setup can for sure overcome spending more hardware resouces documented in IBM DS8900 series spec paper.

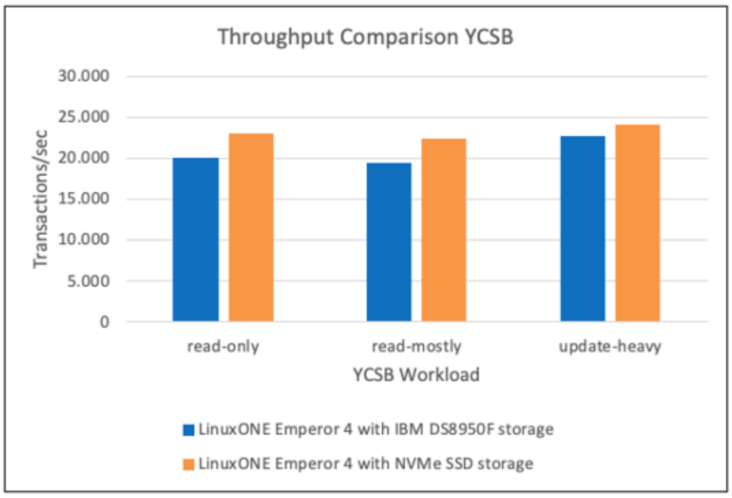

Figure 7: PostgreSQL throughput comparison of different YCSB workload pattern

The throughput rate when running the BDI benchmark on IBM Db2 Warehouse 11.5.7.0-cn3 shows an advantage for the FCP attached LUNs on an IBM DS8950F server with flash card set storage for both memory configurations. With the BDI benchmark the read/write rates are not constant as in case of the YCSB benchmark on PostgreSQL 14.1 but shows burstiness with high write rates up to 3.3 GB/s for a short period of time and low read rates. Since, the IBM DS8950F server with flash card set storage has an huge internal cache of 3.84 TB, those spikes can be buffered very well while the local NVMe RAID-10 suffers to buffer those spikes despite their high write speed visible in I/O wait rates >6%. This is the reason for the small advantage of the IBM DS8950F server with flash card set storage observed in Figure 8.

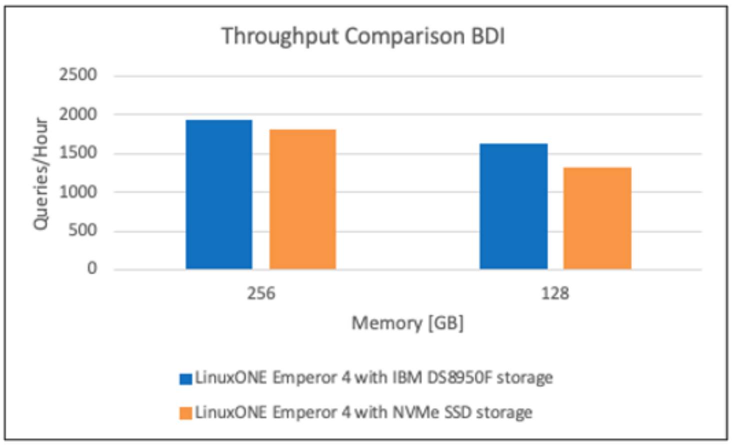

Figure 8: IBM Db2 Warehouse throughput comparison of different BDI measurement

4.1.2 Conclusion

For these workloads we can state that

- If read I/O requests are completely saturating adapters and storage system as we observe with the YCSB benchmark on PostgreSQL 14.1 causing a big number of small I/Os, the workload performance advantage depends upon the number of I/O requests that the storage can process at a time. Therefore, the number of direct-attached NVMe SSDs and the specific RAID setup performance versus the number of used I/O adapters and connections to the external storage server are of most importance.

- If write I/O request are occurring in bursts like we observe with BDI benchmark on IBM Db2 Warehouse 11.5.7.0-cn3, the internal cache of the external storage server shows superior buffering capabilities and we get up to 7% more analytic queries processed during the test run compared to the local NVMe RAID setup.

4.2 Price-Performance

This section shows the price-performance comparisons for the executed workloads on the system under test.

4.2.1 Disclaimer

Performance and price-performance result is extrapolated from IBM internal tests running the

- YCSB benchmark workloads read-only, read-mostly, update-heavy on PostgreSQL on IBM® LinuxONE Emperor 4 native LPAR using direct-attached NVMe SSD storage versus FCP attached LUNs on an IBM DS8950F server with flash card set storage and purchase costs for the measured storage options. Storage costs were extrapolated from purchase costs for 4× 1.92TB NVMe SSDs and 4x IBM NVMe Carrier versus the purchase costs for an IBM DS8950F server with flash card set storage with 25.6 TB storage.

- BDI benchmark, which is based on TPC-DS, on IBM Db2 Warehouse on IBM® LinuxONE Emperor 4 native LPAR using direct-attached NVMe SSD storage versus IBM DS8950F server with flash card set storage and purchase costs for the measured storage options. Storage costs were extrapolated from purchase costs for 4× 1.92TB NVMe SSDs and 4× IBM NVMe Carrier versus the purchase costs for an IBM DS8950F server with flash card set storage.

Results may vary.

4.2.2 Observations

The price-performance with the YCSB benchmark on PostgreSQL 14.1 is

- up to 96% better

using direct-attached NVMe SSD storage compared to using FCP attached LUNs on an IBM DS8950F server with flash card set storage connected by 2 FICON Express32S adapters.

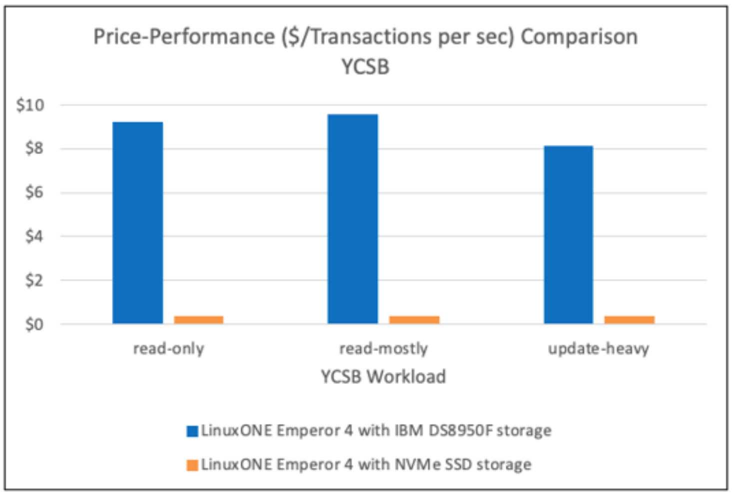

Figure 9: IBM Db2 Warehouse price-performance comparison of different YCSB workload pattern

The price-performance running the BDI benchmark on IBM Db2 Warehouse 11.5.7.0-cn3 is

- up to 94% better

using direct-attached NVMe SSD storage compared to using FCP attached LUNs on an IBM DS8950F server with flash card set storage connected by 2 FICON Express32S adapters.

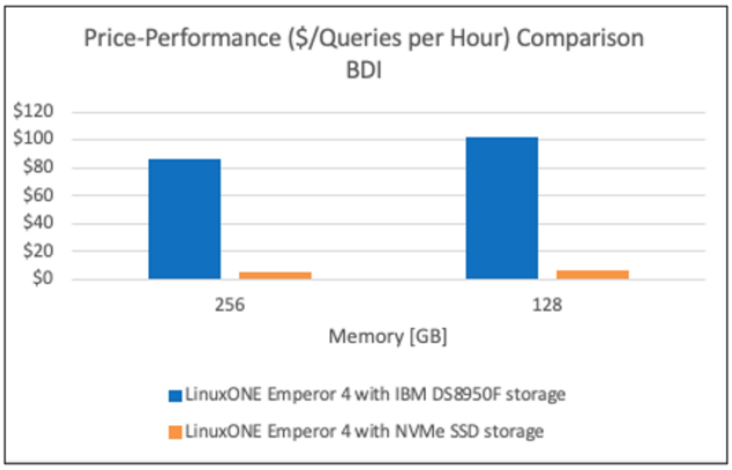

Figure 10: IBM Db2 Warehouse price-performance comparison of different BDI measurement

4.2.3 Conclusion

For these workloads we can state that NVMe SSD storage has a compelling price-performance advantage over I/O attachments with storage servers. The NVMe SSDs in RAID-10 configuration deliver great throughput but lack the high availability features and huge storage capacity (limited to 16 devices, where the raw capacity is reduced by RAID setup) compared to a IBM DS8950F server with flash card set storage. The IBM DS8950F server with flash card set storage provides superior flexibility and high availability capabilities compared to the NVMe SSDs. Therefore, we would recommend the use of NVMe for temporary (buffered) data or databases used as caches that benefit from a high read throughput.

5 References

The following documents are referenced in this white paper:

IBM® LinuxONE Emperor 4

- https://www.ibm.com/it-infrastructure/linuxone
- https://www.ibm.com/products/linuxone-emperor-4

NVMe specifications

- https://nvmexpress.org/resources/specifications

IBM Storage and the NVM Express Revolution (IBM Redbooks®)

- https://www.redbooks.ibm.com/redpapers/pdfs/redp5437.pdf

82037982-USEN-01