Linux on Z and LinuxONE

Scaling HyperPAV alias devices on Linux guests on z/VM

Note: Before using this information and the product it supports, be sure to read the general information under “Notices” on page 17.

Contents

Figures

About this publication

This study analyses the performance improvements of real and virtualized disk setups when scaling HyperPAV alias devices for different workload levels.

Author

Dr. Juergen Doelle

Remarks

The web links referred to in this paper are up to date as of April 06, 2020.

Chapter 1. Introduction

DASD or ECKD devices are commonly used types of disk in IBM Z® environments. They are easy to manage and provide a lot of functionality. For example, on Linux on Z, the DASD device driver already provides high availability through multipathing and load balancing over the available paths without any further setup in the Linux stack. Additionally, special types of processors (SAPs) are implicitly used to execute the I/O operations, saving computation cycles on the Integrated Facility for Linux (IFL) processors for the execution of the workload.

A disadvantage from the performance point of view is that the underlying channel subsystem can execute only a single I/O operation (to be precise: one channel program) on a single device at one point in time. This restriction leads to multiple I/O operations on the same disk being serialized within the operating system. The contention can be shown by monitoring the DASD channel queue in Linux, for example with the DASD statistics. For details about DASD statistics and the tunedasd command, see the Device Drivers, Features, and Commands book at https://www.ibm.com/support/knowledgecenter/linuxonibm/liaaf/lnz_r_dd.html.

The impact is small on a normal Linux root device, but it can become a severe bottle neck, for example on disks where databases store their data and access it with several I/O processes in parallel (see, for example Oracle Database on Linux on System z - Disk I/O Connectivity Study at https://www.ibm.com/support/knowledgecenter/linuxonibm/liaag/l0orac00_2012.htm).

A container environment setup with a single DASD device holding the cluster file and the container overlay file systems motivated this study. Such setups for Docker, PodMan, or CRI-O (with Kubernetes) environments are prone to I/O access limitations. The disk can be, for example, the root file system of the hosting node.

Such a single-disk setup can lead to a high number of parallel disk I/O operations on that disk. The number of disk operations might scale with the number of containers:

- When micro services are used, large numbers of containers with a very short lifecycle are potentially created continuously.
- When just starting many containers at the same time, for example after some maintenance.
- When pods or containers are updated at the same time.

This contention applies to any scenario running multiple, parallel disk I/O requests on one disk. The impact scales with the degree of parallelism.

A very convenient and effective way to overcome that limitation is the usage of HyperPAV alias devices. They can be shared between LPARs, unfortunately they cannot be shared within one Hypervisor, neither between guests nor between the Hypervisor and a guest. When attaching the devices directly to the guest, the amount of HyperPAV alias devices needed scales with the number of guests, until it is no longer practicable. In these scenarios using a virtualized HyperPAV environment might be an option.

This study investigates the behavior of a complex disk I/O load with workload generators running in parallel on a single disk device when scaling the number of HyperPAV aliases in a z/VM® guest. Additionally, the z/VM feature of virtualized HyperPAV alias devices with a full-pack minidisk is compared to direct-attached real devices on a guest.

Chapter 2. Summary

This study analyses the performance improvements of real and virtualized disk setups when scaling HyperPAV alias devices for different workload levels.

For the setup with real devices, the disks and HyperPAV alias devices are directly attached to the guest. This configuration shows the best performance. The usage of a small number of HyperPAV alias devices, like five, improves the performance by several factors. In the best case an improvement by factor eight was achieved compared to the scenario without HyperPAV alias devices. However, because directly attached HyperPAV alias devices cannot be shared, neither between guests nor between a guest and z/VM, this is useful only for a limited number of guests. Ideally, only for guests with very high disk I/O bandwidth requirements.

Using HyperPAV alias devices significantly improves the throughput on all load levels of the workload, but at higher load levels this incurs additional CPU cost.

An overall approach would be to set up such an environment with 5 to 10 HyperPAV aliases as default. When a higher I/O bandwidth is needed, about 20 HyperPAV aliases are a good approach. Be aware that it is possible to define too many alias devices, which in the worst case might lead to a throughput decrease compared to the scenario with fewer alias devices.

The usage of virtualized devices is an alternative, which simplifies the setup and the management significantly at a moderate impact on throughput and CPU cost. The behavior is very similar compared to the setup with real devices and the overhead of 5% - 10% is within the expected range for virtualization.

The virtualized setup with full-pack minidisks and virtual HyperPAV alias devices can be easily applied to a large number of guests. In this scenario it is only important to have sufficient real HyperPAV alias devices attached to the z/VM.

Note that Linux uses the HyperPAV aliases in a balanced way. In a virtualized disk environment z/VM should therefore have a significantly higher number of real alias devices than the guest with the most virtual alias devices, to reduce the contention on the real HyperPAV alias devices.

z/VM 7.1 and 6.4 supports HyperPAV alias devices attached to the system for paging. Enable HyperPAV devices for paging through the FEATURES ENABLE PAGING_ALIAS system configuration file statement or the CP command SET PAGING ALIAS ON.

Chapter 3. Setup

This section describes the hardware and software used and **how to** setup directly attached or virtualized disk and HyperPAV alias devices.

Hardware

The test environment runs on an LPAR on an IBM z14® (3906-M04) with 16 cores (IFL), SMT2 enabled, and 64 GiB memory.

The root disk of the guest was a Mod 54 DASD from an IBM® DS8886 Model 985 (2831-985) with 0 – 30 HyperPAV alias devices. This disk is also used as test device. It is connected through:

- 8 FICON® Express16S+ channels with 16 Gbit/s link speed
- High Performance FCION enabled on all paths

Software

The Hypervisor was z/VM V.7.1.0 SLU 1902. The guest uses RHEL 8.1 with 8 CPUs and 8 GiB memory.

The benchmark used was the Flexible I/O Tester (fio) version fio-2.2.6-2-g8549. It might be part of your distribution or can be downloaded from https://github.com/axboe/fio.

Setting up the HyperPAV environment

HyperPAV alias devices can be considered as simply a special type of disks devices. They have a device address which can be attached to a guest. The guest needs to exclude this address from its cio_ignore list (if this is configured) and enable the device like any other disk device. The DASD device driver manages the usage of the alias device automatically, no further action is required from the Linux administrator.

The chzdev tool can be used to apply these changes and create the corresponding udev rules for persistency in one step for each device.

HyperPAV alias devices do not have a dasd<x> name, so they are not represented within /dev.

- The lsdasd command lists alias devices with the name "alias".
- The lsblk command does not list alias devices.
- You can find statistics for HyperPAV alias devices in /sys/kernel/debug/dasd.

HyperPAV alias devices are defined on the storage server per logical control unit (LCU). They can be used for any disk from that LCU, but only from them. You can use the lsdasd command with the -u option to verify that:

```
lsdasd -u
Bus-ID    Name      UID
================================================================================
0.0.b5d5  dasdb     IBM.750000000XK801.b500.d5.0000000000007562000000000000000
0.0.b5da  dasda     IBM.750000000XK801.b500.da.0000000000007562000000000000000
0.0.1000  dasdc     IBM.750000000XK801.b900.14.000000000000eac500000000000000
0.0.1016  alias     IBM.750000000XK801.b900.xx.00000000000000000000000000000000
0.0.1017  alias     IBM.750000000XK801.b900.xx.00000000000000000000000000000000
0.0.1018  alias     IBM.750000000XK801.b900.xx.00000000000000000000000000000000
0.0.1019  alias     IBM.750000000XK801.b900.xx.00000000000000000000000000000000
```

Figure 1. HyperPAV alias devices

The alias devices in the example are only used for DASD devices that fully match the highlighted part **IBM.750000000XK801.b900**. That means alias devices with UUID IBM.750000000XK801.b900.xx are only used for dasdc and not for dasda or dasdb. The **b900** describes the LCU of the device. For the devices dasda and dasdb, HyperPAV aliases from LCU b500 would need to be added. Also note that the alias devices have an 'xx' after the LCU name, while DASDs have a hex number.

Another challenge is that the address range from an LCU is shared between DASDs and HyperPAV aliases. The more alias devices are defined, the less DASDs can be defined. This limitation can be overcome by using a virtualized disk environment.

All this makes careful management of the directly attached devices essential.

Setup with real devices

To set up a z/VM guest with real devices, the following steps are needed:

1. To assign real DASD and HyperPAV alias devices to a guest, attach the devices to the guest, using, for example:
   `attach<rdev>*`
   where rdev can also be a range, for example: rdev1-rdev2.
   Hint: To reduce the complexity of the guest setup, add a virtual device address
2. Remove the device addresses from the cio_ignore list if one exists and enable the device. To do this persistently, use the chzdev command, for example:
   `# chzdev dasd 1000 -e`
3. To verify the result, use `lsdasd -u`.

Setup with virtual devices

The setup with virtual devices is implemented with a full-pack minidisk and the definition of virtual HyperPAV alias devices. Note that virtual HyperPAV alias devices can only be defined for full-pack minidisks.

To set up a z/VM guest with virtual devices, the following steps are needed:

1. Define the full-pack minidisk using starting cylinder 0 and the address of the real device in the z/VM user directory as
   `MDISK <vdev> 3390 DEVNO <rdev> MR`
   which enables it under address <vdev> when the guest is started.
2. Remove the device addresses from the cio_ignore list if one exists and enable the device as in step 2 above, but use <vdev> address. Accessing the <volid> directly from Linux destroys the minidisk structure.

Before virtual HyperPAV alias devices can be defined for the guest, at a minimum the same number of real HyperPAV alias devices need to be attached to the z/VM. If multiple guests should use virtual HyperPAV alias devices, consider assigning more real devices to the z/VM.

To define virtual HyperPAV alias devices for the guest, the following steps are needed:

1. Attach real HyperPAV alias devices to z/VM using, for example:
   `att $rdev SYSTEM`
   Alternatively, use the SYSTEM_ALIAS statement in the system configuration file.
2. Define the virtual HyperPAV alias devices to the guest:
   `define hyperpavalias <vdev> for base <minidisk address from above>`

The vdev of the HyperPAV alias devices can be any unused address in the guest context. Like the minidisk virtual address, these addresses could be the same for any guest, which simplifies the setup significantly, because the virtual HyperPAV definitions can be identical for all guests.

The next step, using chzdev to remove the device from cio_ignore and enabling the device, is the same as with the real device, except now with the <vdev> address of the HyperPAV alias device.

Note: Neither extended address volumes nor HyperPAV alias devices are eligible for minidisk caching.

Chapter 4. Workload

As workload generator the Flexible I/O Tester (fio) was used. Each fio job starts 8 processes with a mix of:

- read and write operations
- with a varying block size (4K, 8K, 64K, 128K)
- on a large number of files with a varying file size from 1 KB - 100MB

These requests are started asynchronously and with direct I/O.

To scale the workload, the number of jobs was scaled from one to four. All jobs use the same large DASD or minidisk as target

The expectation is that disk I/O workload levels caused by containers and their overlay files systems are between the load level created by one or two fio jobs (which is low to moderate). The level created by four fio jobs can be considered as very high.

The objective was to provide a more complex I/O pattern than just sequential reads or writes to simulate a more customer-like workload.

Chapter 5. Results

This chapter describes the results of scaling real and virtual HyperPAV aliase devices at different workload levels.

All results are averages from two runs.

Directly attached disk and HyperPAV alias devices

This section covers the results when using a real device that is directly attached to the guest.

Single Workload generator (fio job)

This section shows the results with a single fio work generator. Each fio jobs starts 8 processes that issue their workload asynchronously to ensure a certain amount of parallel I/O requests to the test device.

With this workload level, the throughput increases almost proportionally up to five HyperPAV alias devices. With more alias devices, the gain in throughput slowly decreases up to 30 alias devices, where the curve flattens out and the throughput reaches maximum.

Throughput

Figure 2 on page 11 shows the scaling of the throughput when scaling the amount of HyperPAV aliases from none to 30.

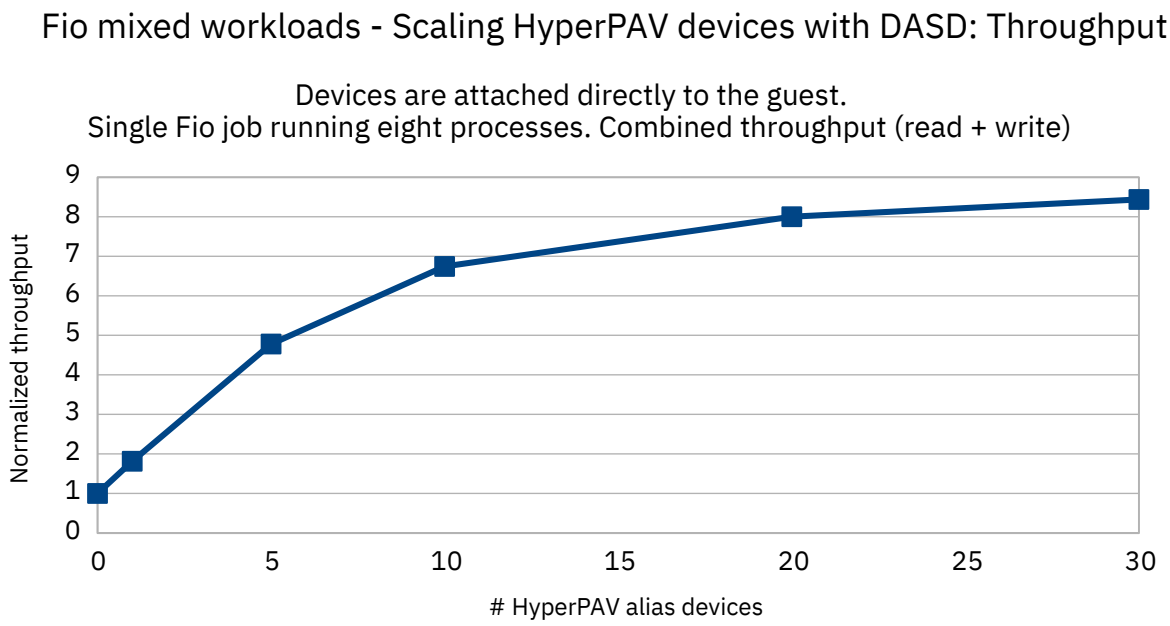

Figure 2. Directly attached devices: Scaling HyperPAV alias devices with a single fio workload generator job – Throughput, normalized to 1 fio job with no HyperPAV alias device.

With this workload level, the throughput increases almost proportionally up to five HyperPAV alias devices. With more alias devices, the gain in throughput slowly decreases up to 30 alias devices, where the curve flattens out and the throughput reaches maximum.

CPU cost

Figure 3 on page 12 shows the scaling of the cost when scaling the number of HyperPAV aliases from none to 30 devices. The cost is defined as throughput driven per CPU, higher values are better.

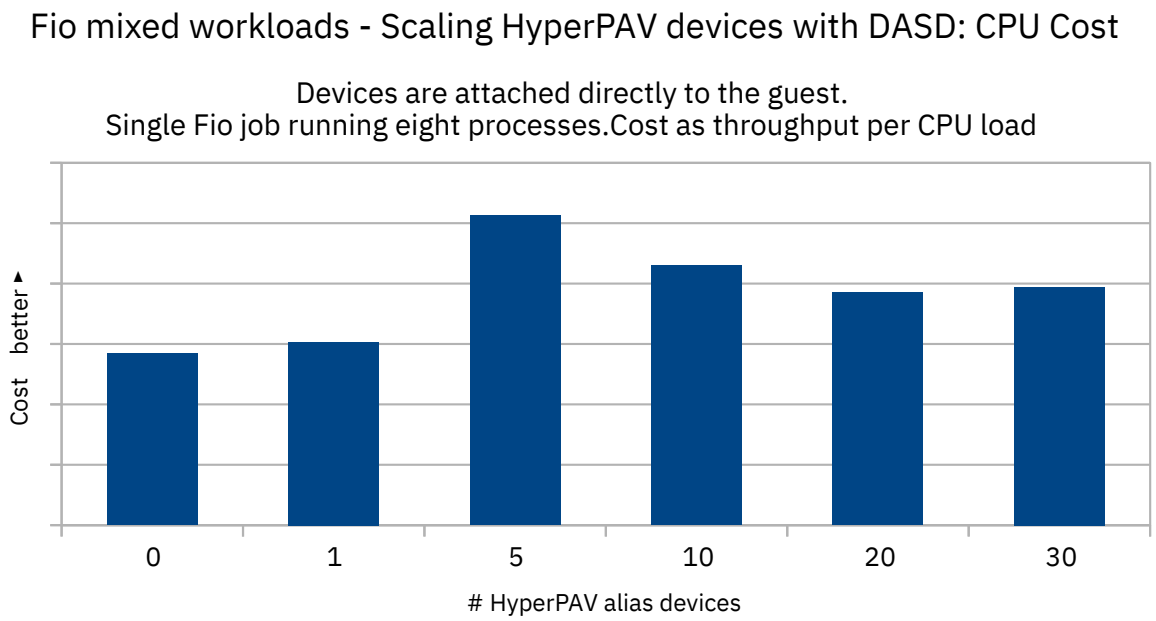

Figure 3. Directly attached devices: Scaling HyperPAV alias devices with one fio workload generator job – CPU cost.

Compared to the setup without alias devices, all scenarios with alias devices show a reduction of the CPU cost leading to an increase of throughput driven per CPU. This workload level works most efficiently (highest throughput value per CPU) with five alias devices. Note that the throughput further increases greatly from five to 30 HyperPAV alias devices, but the consumption of CPU cycles per MB/sec to drive that high level of throughput is higher compared to five alias devices, but still less compared to no HyperPAV aliases.

Conclusion

Using HyperPAV alias devices improves the throughput of low to moderate workloads significantly and always drives more throughput with the CPU cycles than without alias devices.

Scaling the Workload

For this scenario the workload was increased to two and four fio jobs which results in up to 32 workload generating processes. Especially the last workload level could be considered as very high.

Throughput

Figure 4 on page 13 shows the normalized throughput when scaling the workload in three steps by scaling the number of fio jobs.

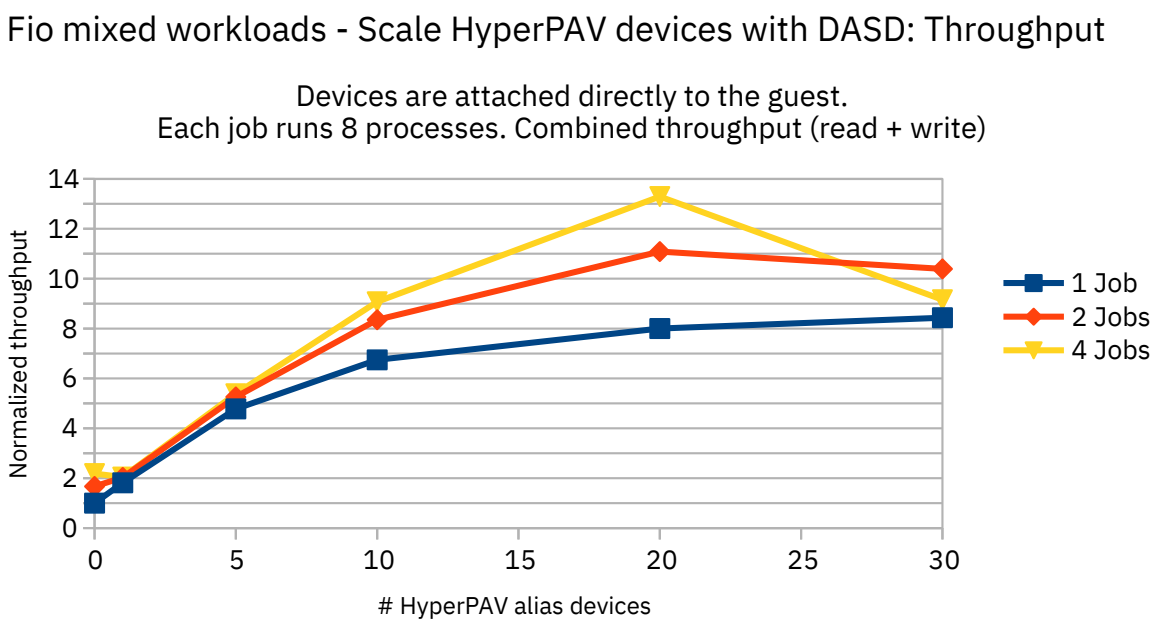

Figure 4. Directly attached devices: Scaling HyperPAV alias devices and fio workload generator jobs – Throughput (normalized to 1 fio job with no HyperPAV alias device).

There are two important aspects:

- The throughput scaling with a fixed number of HyperPAV alias devices.
  - When increasing the load level, the throughput only scales with 20 alias devices for all three workload levels. With ten alias devices it only scales well from one to two fio jobs.
  - This shows the contention caused by serialization effects.
- The throughput scaling when scaling the number of HyperPAV alias devices for a certain workload
  - Each workload level improves with more HyperPAV alias devices up to 20 devices.
  - With increasing numbers of HyperPAV alias devices, more bandwidth of the I/O subsystem becomes available to handle parallel I/O requests to the same disk.

When scaling the workload level, the improvement already seen with the low workload level by using alias devices continues. It reaches an improvement up to factor 8x (!) when comparing 20 alias devices with no alias devices for the same workload level.

But it also shows a saturation effect for more than 20 alias devices, where the throughput curve flattens or declines at the end when increasing the amount of alias devices. The effect is larger for higher workload levels.

CPU cost

Figure 5 on page 14 compares the CPU cost as throughput per CPU. It shows the ratio of the cost results with HyperPAV alias devices versus without alias device (value with no alias device defines the base line).

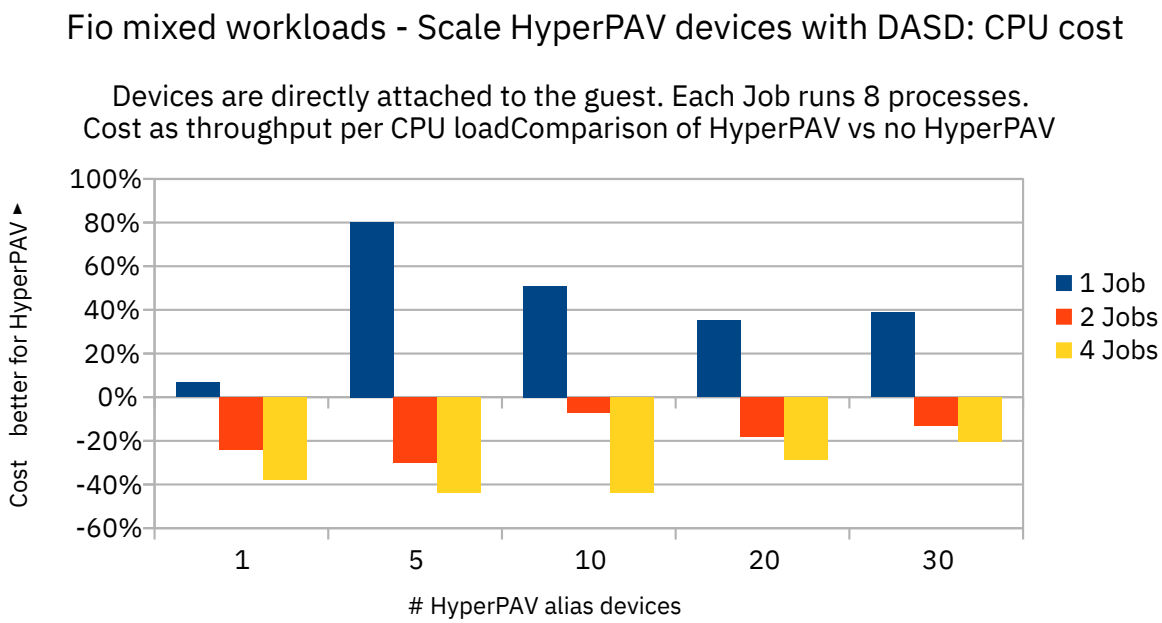

Figure 5. Directly attached devices: Scaling HyperPAV alias devices and fio workload generator jobs – Comparison of CPU utilization cost (throughput/CPU load) with HyperPAV alias devices versus without.

Compared with a single fio job results the picture changes. The blue bars repeat what is shown in figure 2 above for one fio job. With increasing workload, the effort for driving the alias devices increases. However, the throughput improves by factors while the cost increases in percent. For example, you get more than 6x -8x more throughput at a cost (CPU effort per throughput) increase of 20% to 40%.

Conclusion

Using HyperPAV alias devices improves the throughput on all workload levels significantly, but at higher workload levels the improvement incurs some additional CPU cost.

Overall, a useful approach is to set up such an environment with 5 – 10 HyperPAV aliases. If a higher I/O bandwidth is required, 20 HyperPAV aliases are a good approach. Be aware that it is possible to have too many alias devices defined. In the worst case this can lead to a decreased throughput compared to the best case.

Minidisk and virtual HyperPAV alias devices

The virtual environment of the guest consists of full-pack minidisks and virtual HyperPAV alias devices created with the **define hyperpavalias** statement. Now, z/VM owns the same number of real HyperPAV alias devices.

A major difference when using the virtualized setup is the increase in path length. z/VM now drives and manages the virtual devices additionally to the I/O to the real devices. Due to this additional workload the important question here is the impact on throughput and CPU cost.

Overall the behavior is very similar to the directly attached devices. Therefore, the focus of the following figures is on what changes compared to the environment with real devices. To depict that, the charts show only the ratio of the minidisk data to the corresponding DASD data. Positive values mean that the minidisk shows a better behavior (higher throughput or lower CPU cost) than the DASD.

Throughput

Figure 6 on page 15 shows the ratio of the minidisk throughput versus the DASD value when scaling virtual HyperPAV aliases devices.

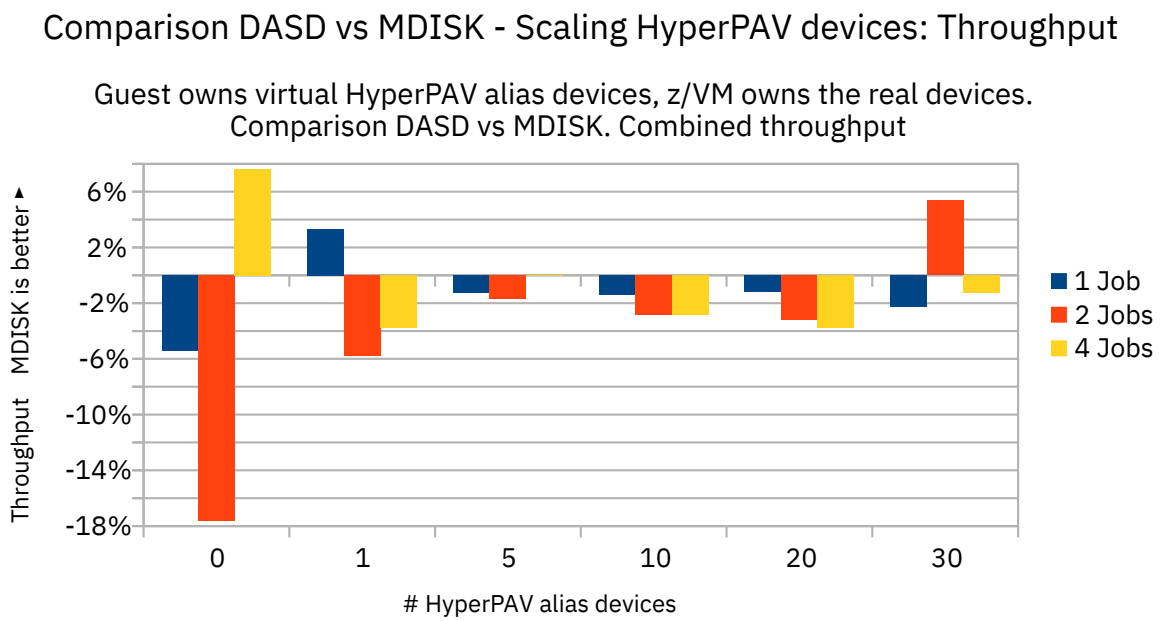

Figure 6. Virtual devices: Scaling HyperPAV alias devices and fio workload generator jobs – Ratio of throughput with the virtual devices versus attached real devices.

Overall, the throughput shows a moderate degradation from -1% to -4%. The scenario without HyperPAV aliases behaves differently, here all values are relatively small, meaning that the -18% degradation corresponds to an absolute difference of 50 MB/sec. The improvement to the setup without virtual HyperPAV aliases is still up to a factor of 5.4x-8.7x.

CPU cost

Figure 7 on page 16 show the ratio of the minidisk CPU cost versus the DASD value when scaling virtual HyperPAV aliases.

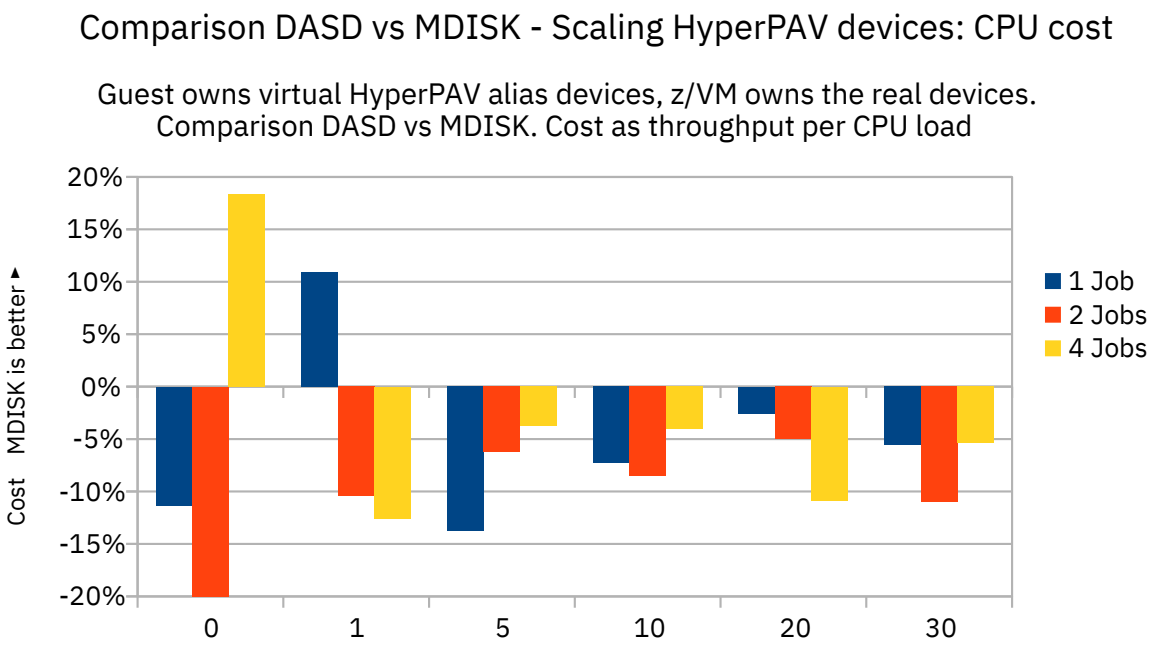

Figure 7. Virtual devices: Scaling HyperPAV alias devices and fio workload generator jobs – Ratio of the CPU cost with the virtual devices versus attached real devices.

Overall the CPU cost shows an additional effort for the virtualization from 5% to 10% (average 6%), which is in the expected range for a virtualized setup.

Conclusion

Because it is not possible to set up a larger number of z/VM guests with directly attached HyperPAV alias devices, as the real devices cannot be shared between guests, the virtualized setup is a good alternative. The overall behavior, with its moderate impact on throughput and cost, is similar compared to the setup with directly attached real devices. The overhead is almost independent of the number of alias devices.

The virtualized setup with full-pack minidisks and virtual HyperPAV alias devices can easily be applied to large numbers of guests. It is only important to have enough real HyperPAV alias devices attached to the z/VM.

- The minimum requirement is the number of devices of the guest with the most virtual HyperPAV alias devices. Configuring a guest with more virtual alias devices than the z/VM system has real alias devices will fail.
- The virtualized HyperPAV alias devices from the guests are dispatched to the real HyperPAV alias devices from z/VM. Therefore, to reduce contention, the z/VM should be configured with a significantly larger number of real HyperPAV alias devices than the largest guest has.

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing 2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Software Interoperability Coordinator, Department 49XA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information in softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Terms and Conditions

Permissions for the use of these publications are granted subject to the following terms and conditions.

Personal Use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative works of these publications, or any portion thereof, without the express consent of the manufacturer.

Commercial Use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of the manufacturer.

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any data, software or other intellectual property contained therein.

The manufacturer reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by the manufacturer, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

THE MANUFACTURER MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THESE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

References

The following documents are referenced in this white paper:

1. "Linux on Z and Linux One", Device Drivers, Features, and Commands Development stream (Kernel 4.19)
   https://www.ibm.com/support/knowledgecenter/linuxonibm/liaaf/lnz_r_dd.html
2. "Oracle Database on Linux on System z® - Disk I/O Connectivity Study"
   https://www.ibm.com/support/knowledgecenter/linuxonibm/liaag/l0orac00_2012.htm

Printed in USA